Table of content

0.0 Introduction to part B **6**

1.0 Source Code **7**

1.1 Project Structure 7

1.2 Key Classes Code Snippets 9

2.0 Test Cases / User Testing Evidence **12**

2.1 Overview of Test Cases 12

2.1.1 Purpose of This Document: 13

2.1.2 Explanation of Columns: 13

2.2 Evidence of Testing 14

TC01: Login as Customer with valid credentials 14

TC02: Login with unregistered email 16

TC03: Reset forgotten password 17

TC04: Register 19

TC05: Submit Issue 22

TC07: Chatbot 24

TC09: Admin Login 26

TC10: Manage Tickets 28

TC11: Update FAQ 30

2.3 UI Wireframes Referenced in Test Cases 32

2.4 Wireframe Descriptions 35

3.0 User Guide / README **37**

3.1 Introduction 37

3.2 User Accounts 37

3.3 System UI Walkthrough 38

3.3.1 Customer Interfaces 38

3.3.1.1 Login Page 38

3.3.1.2 Customer Login Validation 39

3.3.1.3 Customer Dashboard 40

3.3.1.4 Submit Issue 41

3.3.1.5 Track My Request 42

3.3.1.6 Ticket Details 43
3.3.1.7 FAQ Help 44
3.3.1.8 Chatbot 45
3.3.2 Admin Interfaces 46
3.3.2.1 Admin Login 46
3.3.2.2 Admin Dashboard 47
3.3.2.3 Manage Tickets 48
3.3.2.4 View Ticket 49
3.3.2.5 FAQ Update 50
3.3.2.6 Add New FAQ 51
4.0 Conclusion **52**
5.0 Rubrics **53**
6.0 Appendix **53**
Appendix A: Wireframe 53

0.0 Introduction to part B

This report is the last phase of the Food Delivery Customer Support project which focuses on the full implementation of the system using JavaFX and Scene Builder. Based on the analysis and design made in the first part of the project, this phase highlights the practical application of OOP and GUI principles in addition to robust event handling mechanisms to build a responsive and user-friendly experience.

This project is developed to ease customer support systems within a food delivery environment. It is designed to address various support needs by offering sets of integrated tools to improve efficiency and accessibility of the support process. which include key features like issue submission, FAQ help page, issue tracking and finally interactive chatbot for assistance. All these functionalities aim to improve customer satisfaction by providing accessible and efficient support tools.

The development process was carried out with a strong focus on maintainability and scalability. Scene Builder plays a great role in this process where it is used to simplify the interface designing process through a visual drag-and-drop feature. By using FXML to define the layout and pairing it with java controller classes, the project gains benefit from clean separation of concern, thus boosting the clarity and organization of the code.

This report covers the implementation process in detail and includes the complete overview of the system architecture and explains how the different components interact with each other to deliver the intended functionality. The full complete source code is included and organized into respective well defined packages. Besides the code, this report also highlights the relevant testing procedure to validate the system. This is to ensure the system’s functionality and reliability.

Lastly, to assist end-users, a comprehensive user guide is also included within this report. It explains how to navigate through the system and how to use each functionality. Where appropriate, diagrams are used to visually demonstrate the application’s workflow.

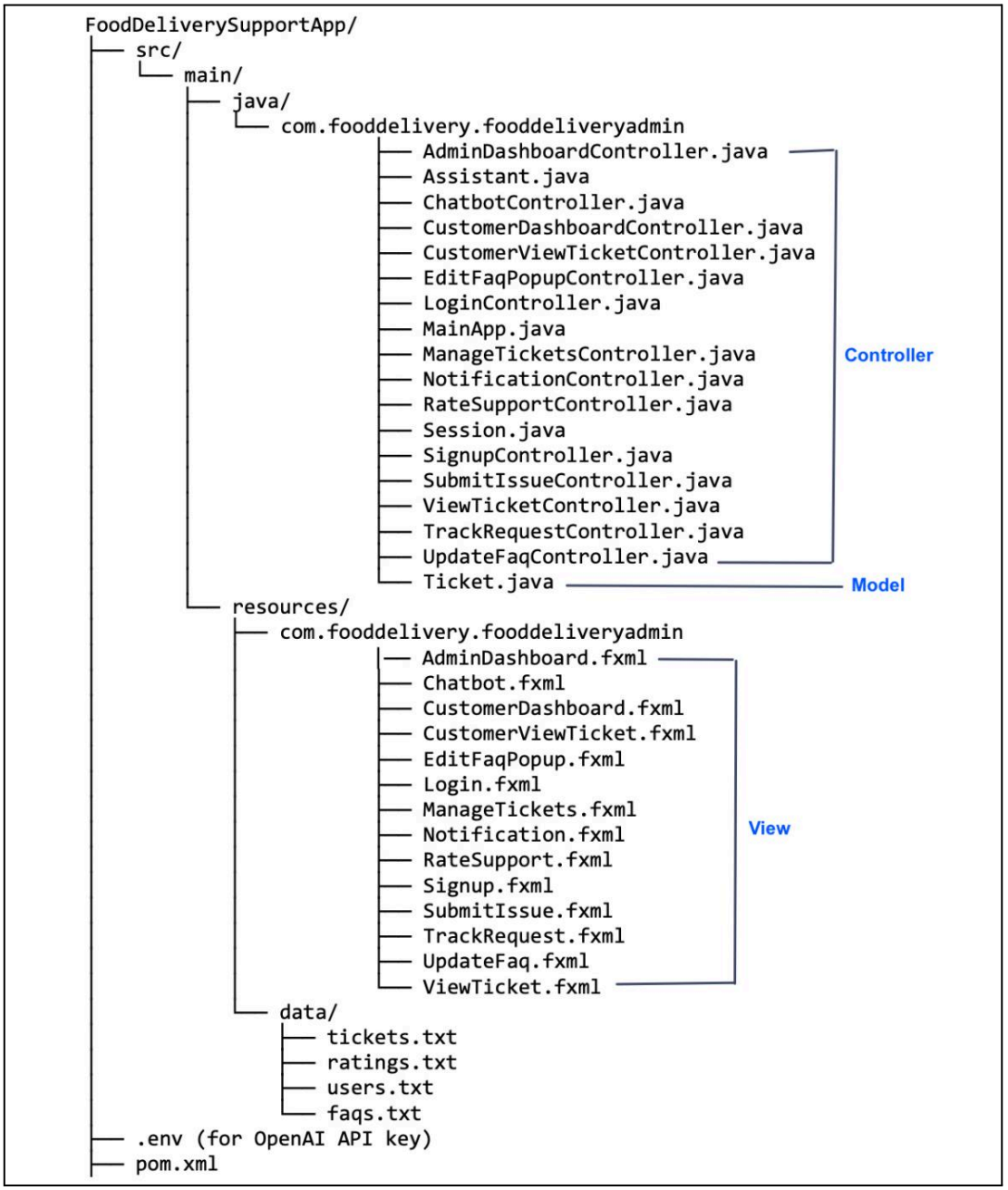
To wrap the introduction up, our team hopes to deliver the useful inputs in this report in order to help understand the last part of the project. Criterias such as readability & comments, data handling, event handling and completeness are ensured to be stressed on throughout the report.

1.0 Source Code

In this project, we used JavaFX along with MVC architecture (Model-View-Controller) to keep everything organized. All the logic, UI, and data are separated so it's easier for us to manage and maintain the app.

1.1 Project Structure

The project is structured inside the *FoodDeliverySupportApp* folder and split into a few parts:



The main parts include:

1. **Controller (Java folder)**

   This is where all the main logic happens. Each *.java* class here controls different screens and features in the app. For example:

- LoginController handles the login process.
- SubmitIssueController lets customers submit issues.
- ManageTicketsController allows admin to view and respond to tickets.
- ChatbotController connects to OpenAI via LangChain4j to simulate a support chatbot.

   In total, there are 18 Java classes here.

2. **Model**

   This part holds our *Ticket.java* class, which acts as the data model for customer support tickets. It uses *SimpleStringProperty* to work smoothly with JavaFX tables.

3. **View (FXML folder)**

   All the user interfaces are designed using *.fxml* files. Each FXML matches a controller. These files define the layout and styling of each screen.

4. **Data Files**

   Instead of using a real database, we store our data in *.txt* files like:

- users.txt: for storing login credentials
- tickets.txt: for all customer support tickets
- ratings.txt: for user feedback

5. ***.env* File**

   We also added a *.env* file to store our OpenAI API key securely, so the chatbot can connect to GPT.

6. ***pom.xml***

   Since the project uses *Maven*, we manage dependencies like *JavaFX, LangChain4j*, and *dotenv* through this file.

1.2 Key Classes Code Snippets

In this part, we briefly explain the main classes that contribute directly to the core functionality of our Food Delivery Support System. These classes show key Java OOP and JavaFX concepts in action like encapsulation, UI control, scene navigation, and even external API integration via *LangChain4j*.

1.2.1 Ticket Model

Ticket.java

```
//Property used for binding tix ID to UI component
private final SimpleStringProperty ticketId;

//Getter
public String getTicketId() { return ticketId.get(); }

//Property accessor
public SimpleStringProperty ticketIdProperty() { return ticketId; }
```

Key Concept: *Encapsulation + JavaFX Property Binding*

We used *SimpleStringProperty* instead of plain String so that the data can automatically update inside JavaFX tables. This is useful for binding data in *TableView*. The class also follows encapsulation, where all attributes are private, and we use getters and setters to access or change the data safely. This represents a standard OOP practice.

1.2.2 Submit Issue

SubmitIssueController.java

```
String ticketId = "#T" + UUID.randomUUID().toString().substring(0,
4).toUpperCase();

//Save ticket data into the text file, create it if needed and appending to it
Files.writeString(TICKET_FILE, line + System.lineSeparator(),
StandardOpenOption.CREATE, StandardOpenOption.APPEND);
```

Key Concept: *UUID for ID + File Storage + Validation*

When a customer submits an issue, we generate a unique ticket ID using *UUID*. This avoids duplicate IDs. Then we save the ticket data into a .txt file using *Files.writeString()*, this works like a mini-database. We also do simple validation to make sure no field is left empty before saving. This helps ensure clean and complete data.

1.2.3 Manage Tickets

ManageTicketsController.java

```
actionCol.setCellFactory(col -> new TableCell<>() {
    private final Button btn = new Button("View");
    {
        btn.setOnAction(e ->
openTicketView(getTableView().getItems().get(getIndex())));
    }
});
```

Key Concept: *TableView + Custom Button (JavaFX GUI)*

We used *TableView* to display all submitted tickets to the admin. Each row has a “View” button, added dynamically using a *CellFactory*. This is a JavaFX GUI concept where we customize each table cell. When the button is clicked, it opens a detailed view of that specific ticket using *FXMLLoader*.

1.2.4 Admin View Ticket

ViewTicketController.java

```
@FXML private void onSetResolved() {
    setStatus("Resolved");
}
```

Key Concept: *Status Control + Quick Reply*

The admin can update the ticket status (Pending, In Progress, Resolved) by clicking the respective button. We use a helper method *setStatus()* to avoid repeating code. We also added some pre-written replies like “Refund has been made...” so admins can respond quickly. This improves user experience and saves time.

1.2.5 Track My Request

TrackRequestController.java

```
//Get currently logged-in user
String currentUser = Session.getCurrentUser();

//Check if tix belong to current user
//If yes, add and display it to user’s tix list
if (email.equalsIgnoreCase(currentUser.trim())) {
    ticketList.add(ticket);
}
```

Key Concept: *Filtering Tickets by User Session*

Customers should only see their own tickets. So, we store the current user's email in a static Session class. Then we filter the ticket file and only show tickets that match the logged-in email. This is like a simple way to simulate session-based access control.

1.2.6 Chatbot

ChatbotController.java

```
OpenAiChatModel model = OpenAiChatModel.builder()
    .apiKey(apiKey)
    .modelName("gpt-3.5-turbo")        //setup openAi model
    .build();

assistant = message -> model.generate(message);  //chatbot assistant logic
```

Key Concept: *LangChain4j + OpenAI Integration*

We added a chatbot to help customers instantly. It uses *LangChain4j* to connect with OpenAI's GPT model using an API key. The chatbot replies based on user questions and uses a VBox to show messages like real chat apps. This feature improves user interaction and showcases integration of AI into our app.

1.2.7 FAQ Management

UpdateFaqController.java

```
List<String[]> faqLines = Files.lines(FAQ_FILE)
     //read line and split into question and answer
    .map(line -> line.split("::", 2))
    .collect(Collectors.toList());
```

Key Concept: *Dynamic GUI + File Update*

The admin can edit FAQs in the app. We load the questions and answers from a .txt file and display them as dynamic cards using JavaFX HBox and VBox. We used *Java Streams* to process the file efficiently. When the FAQ is updated, we write the new content back into the file.

1.2.8 Sign Up + Login

SignupController.java

```
if (!username.contains("@")) {
    username += "@example.com";
}
```

Key Concept: *Role Assignment and Input Checks*

For safety, we only allow sign-ups as customers. The code ensures that all fields are filled and that passwords match. The email is auto-fixed if the user forgets the @ symbol, this is a small UX enhancement to help users. After registration, the new user is added to a text file with *username::password::role*.

1.2.9 Session Management

Session.java

```
public static void setCurrentUser(String username) {
    currentUser = username;
}
```

Key Concept: *Tracking Logged-in User*

This is a simple utility class that stores the currently logged-in user's email. It's marked *static* so we can access it from any controller without passing data manually. This is important for keeping the app aware of who is currently using it, especially for personalized views.

In summary, we can say that each class plays a specific role, from handling tickets and user sessions to building UI screens and chatbot interactions. We applied OOP concepts like encapsulation and modular design, while using JavaFX for event handling and clean interfaces.

The full version of the source code of our Food Delivery Support app can be found in the zip file attached together with this report.

2.0 Test Cases / User Testing Evidence

2.1 Overview of Test Cases

ID	Feature	Test Description	Preconditions	Test Steps	Expected Result
TC01	Login	Verify login with valid credentials	User is registered	1. Go to Login Page 2. Enter email and password 3. Click Login	User is redirected to dashboard
TC02	Login	Verify login with invalid credentials	User is registered	1. Go to Login Page 2. Enter wrong email/password 3. Click Login	Error message is displayed
TC03	Forgot Password	Reset password for user registered	User forgot password	1. Go to Login Page 2.Click “Forgot Password” 3. Enter wrong email/password 4. Click Reset	Password is reset and confirmation shown
TC04	Register	Verify new user registration	User not registered	1. Go to Register Page 2. Enter valid details 3. Click Register	User account is created and redirected to login page
TC05	Submit Ticket	Submit a support ticket	User is logged in	1. Go to Submit Ticket Page 2. Fill all fields 3. Click Submit	Ticket is created and confirmation shown
TC06	FAQ	Search FAQ entries	FAQ entries exist	1. Go to FAQ Page 2. Enter keyword in search bar	Relevant FAQ entries are displayed
TC07	Chatbot	Ask a question to chatbot	User is logged in	1. Go to Chatbot Page 2. Type a question 3. Click Send	Chatbot returns an appropriate answer
TC08	Profile Update	Update user email address	User is logged in	1. Go to Profile Page 2. Change email 3. Click Save	Email is updated in system
TCO9	Admin Login	Verify login using registered admin account	admin account exists	1. Go to Login Page 2. Enter admin email and password 3. Click Login	Admin is redirected to admin dashboard

TC10	Manage Tickets	Admin views submitted tickets	Admin is logged in	1. Go to Admin Dashboard 2. Click Manage Tickets	List of submitted tickets is displayed
TC11	Update FAQ	Admin updates an existing FAQ	Admin is logged in	1. Go to FAQ Management 2. Edit FAQ entry 3. Click Save	FAQ is updated successfully
TC12	Logout	Verify user logout	User is logged in	1. Click Logout in menu	User is redirected to login page

This set of test cases is designed to validate the functionalities of the Food Delivery Support Application. Each test case provides a structured way to check whether specific features of the application are working correctly according to the project requirements.

2.1.1 Purpose of This Document:

- Ensure all functional requirements are met.
- Identify bugs or unexpected behaviors early.
- Help developers and testers verify the correctness and completeness of each feature.

2.1.2 Explanation of Columns:

Column	Description
Test Case ID	A unique identifier for each test case (e.g., TC01, TC02).
Feature	The system module or function being tested.
Test Description	A brief summary of what is being tested.
Preconditions	The required state or setup before performing the test.
Test Steps	The specific actions to be performed.
Expected Result	The result that should occur if the system is working correctly.

2.2 Evidence of Testing

This section will present measures of the conduct of some of the test cases done to show the effective functionality of the system. Although not every test case was performed, the selected ones include the most important user interactions and significant functionality of the application. There are also test cases (e.g. edge cases or repetitive flows) that have not been tested due to the time restrictions and scope limitation, and their behavior was analytically inferred or described within related scenarios.

TC01: Login as Customer with valid credentials

Input: Registered email & correct password

- **Username:** azra@gmail.com
- **Password:** azra123

Expected Result: Redirected to customer dashboard

Actual Result: A success message Login successful as customer was presented and then he was redirected to the dashboard of customer.

Status: pass

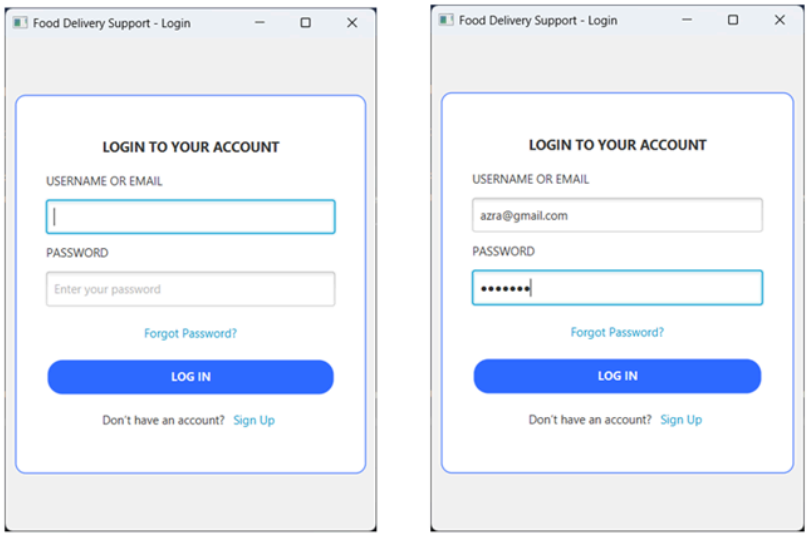


Figure 1: Customer login attempt (before & after input)

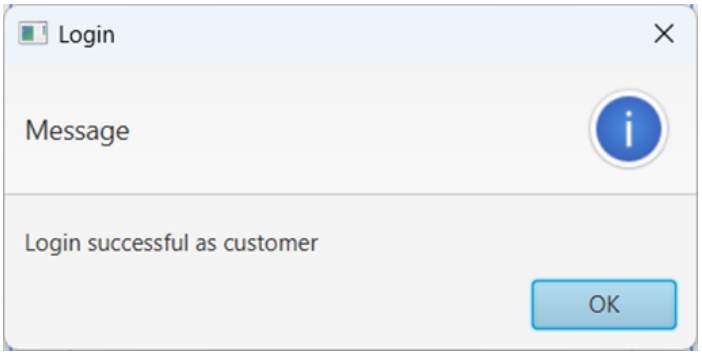


Figure 2: Login success message

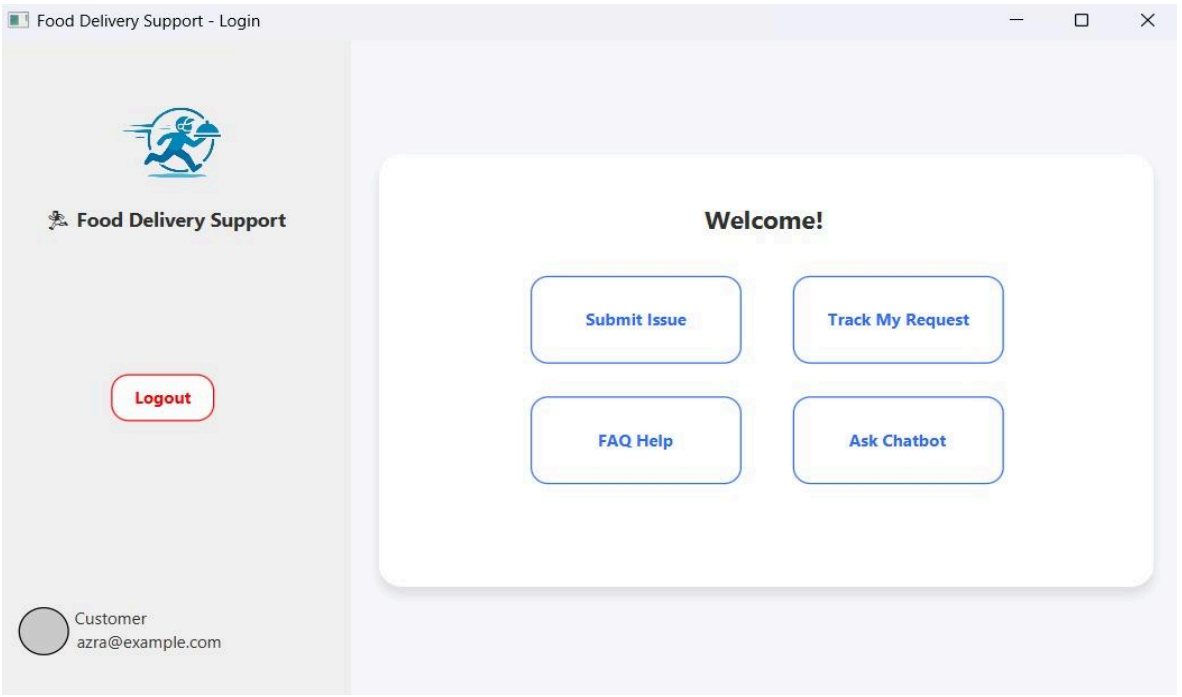


Figure 3: Customer dashboard

TC02: Login with unregistered email

Input: Unregistered email & password

- **Username:** juri@gmail.com
- **Password:** juri123

Expected Result: Redirect to sign up

Actual Result: Error message “Invalid credentials ” was displayed. No redirection occurred.

Status: pass – system handled invalid input with proper error.

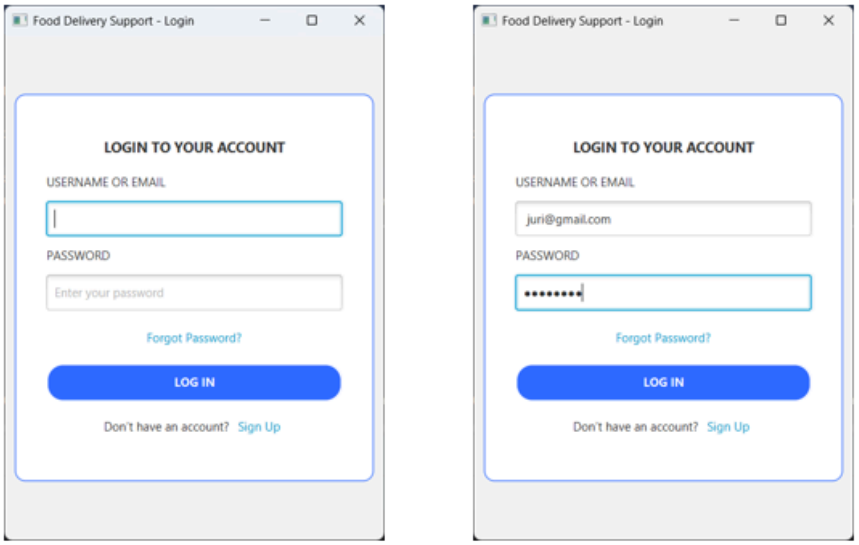


Figure 4: unregistered user login input

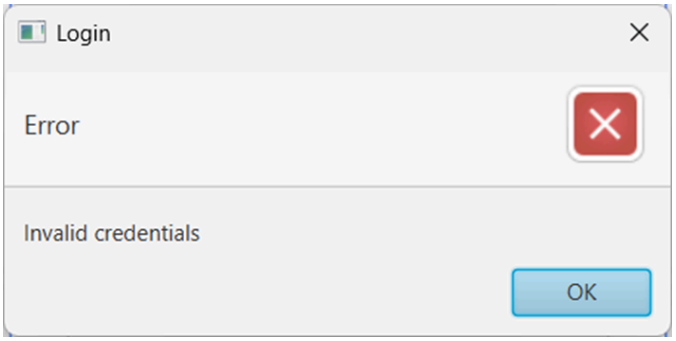


Figure 5: Error message “Invalid credentials ”

TC03: Reset forgotten password

Input: Register email with forgotten password

- **Username:** azra@gmail.com
- **Password:** Forgotten password

Expected Result:

- Successful reset of password and message to confirm the reset is displayed.
- User can then log in with a new password.

Actual Result:

- System displayed“Password reset successfully”
- User was able to login successfully and was redirected to the dashboard to the customer.

Status: pass

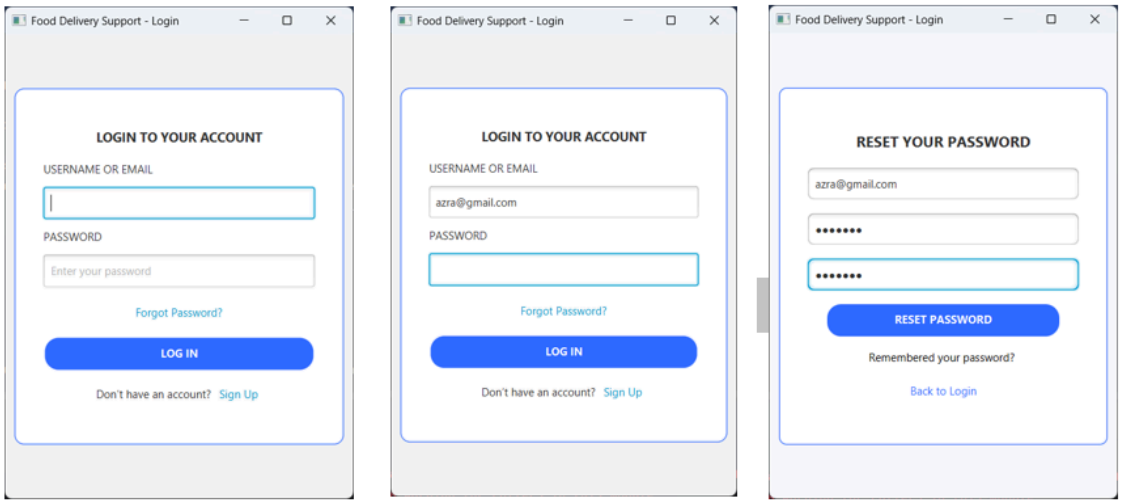


Figure 6: Forgot password flow

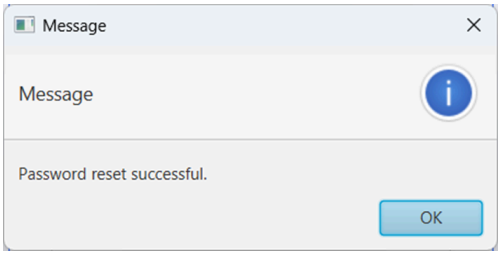


Figure 7: Reset confirmation

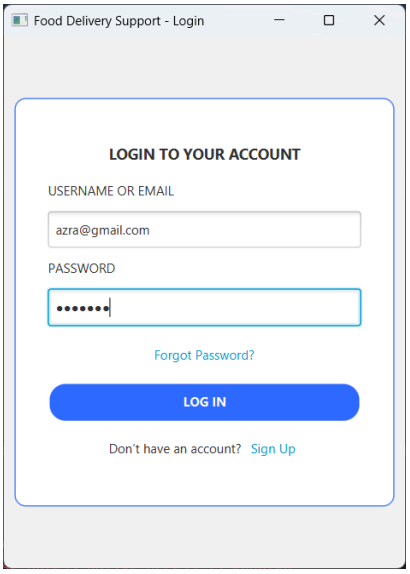


Figure 8: Login with new password

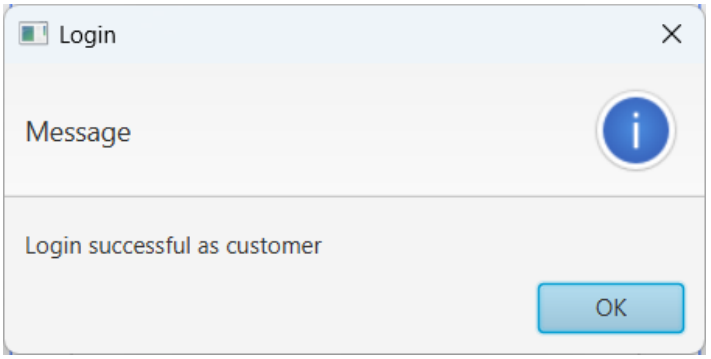


Figure 9: Login success

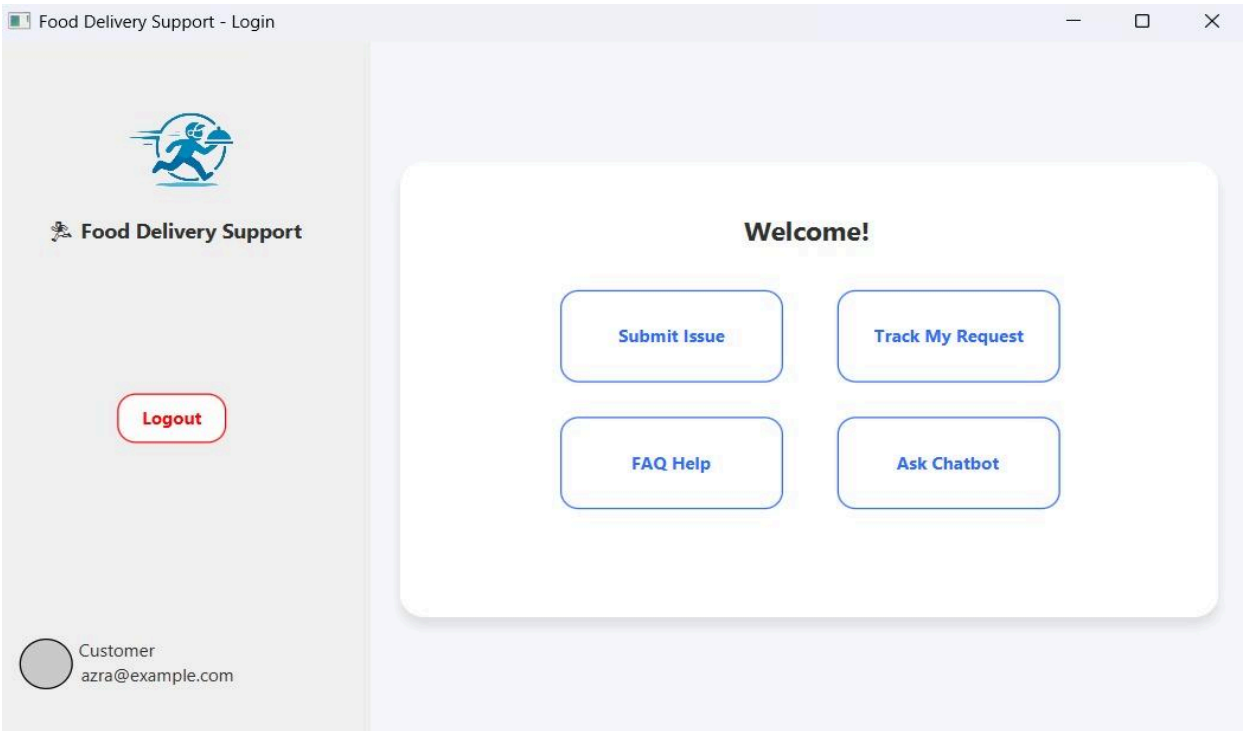


Figure 10: Customer dashboard

TC04: Register

Input: Register a new account with valid credentials

- **Username:** blue@gmail.com
- **Password:** blue1957

Expected Result:

- Successful creation of account and a message to confirm the registration is display
- User can then log in using the registered credentials

Actual Result:

- System displayed “Account created successfully!”
- User was able to login successfully and was redirected to the dashboard as customer.

Status: pass

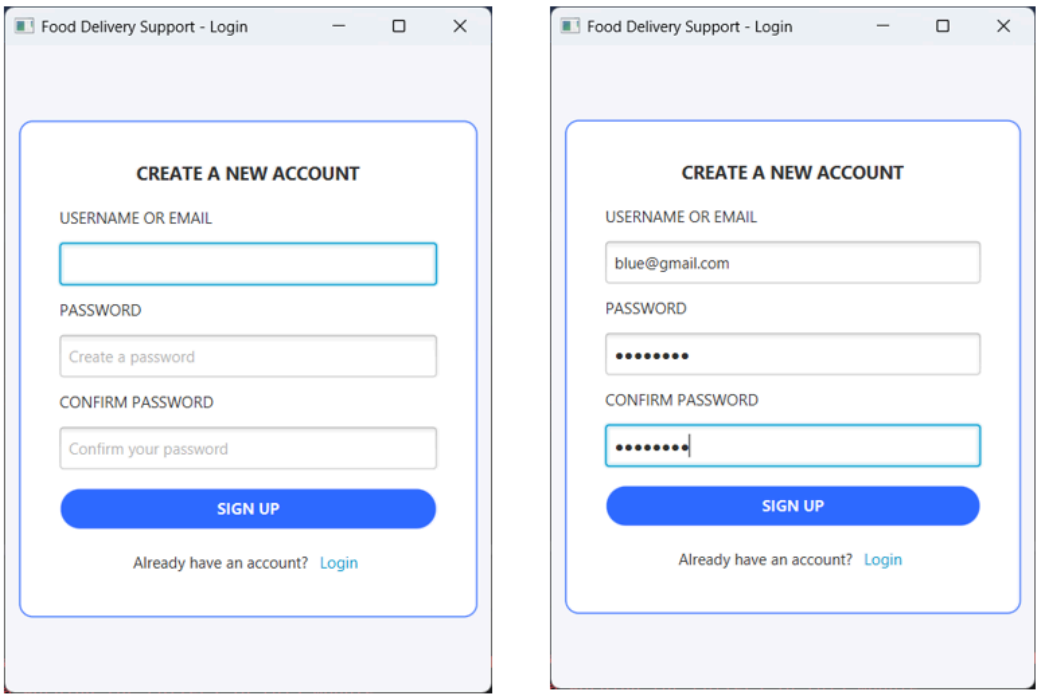


Figure 11: Register new account

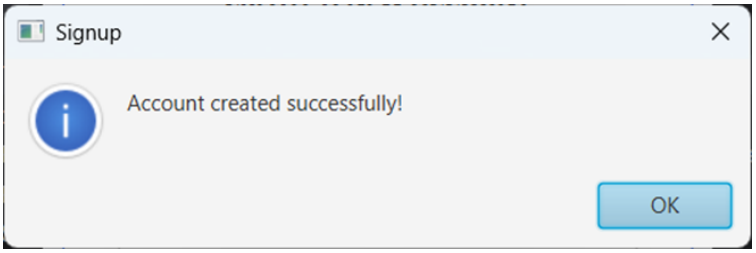


Figure 12: Account created successfully

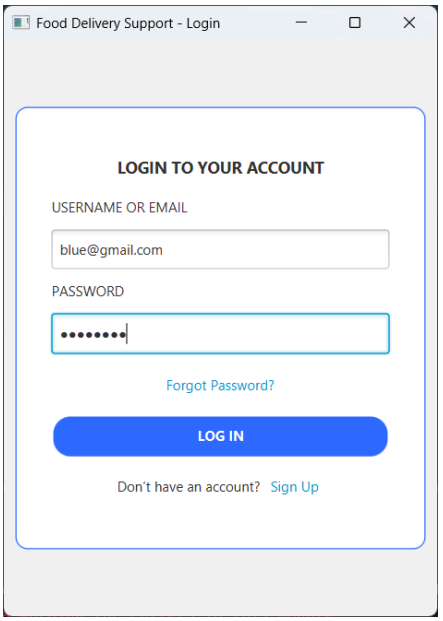


Figure 13: Login with new account

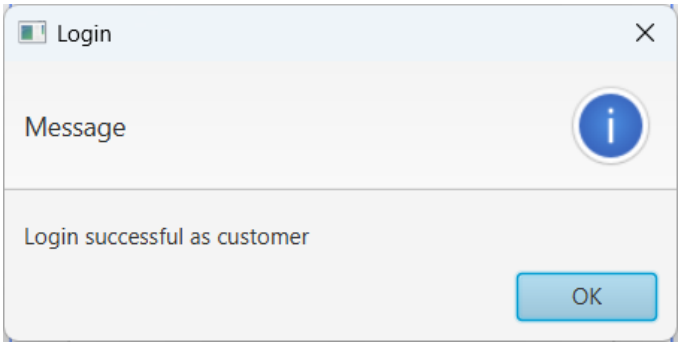


Figure 14: Login success

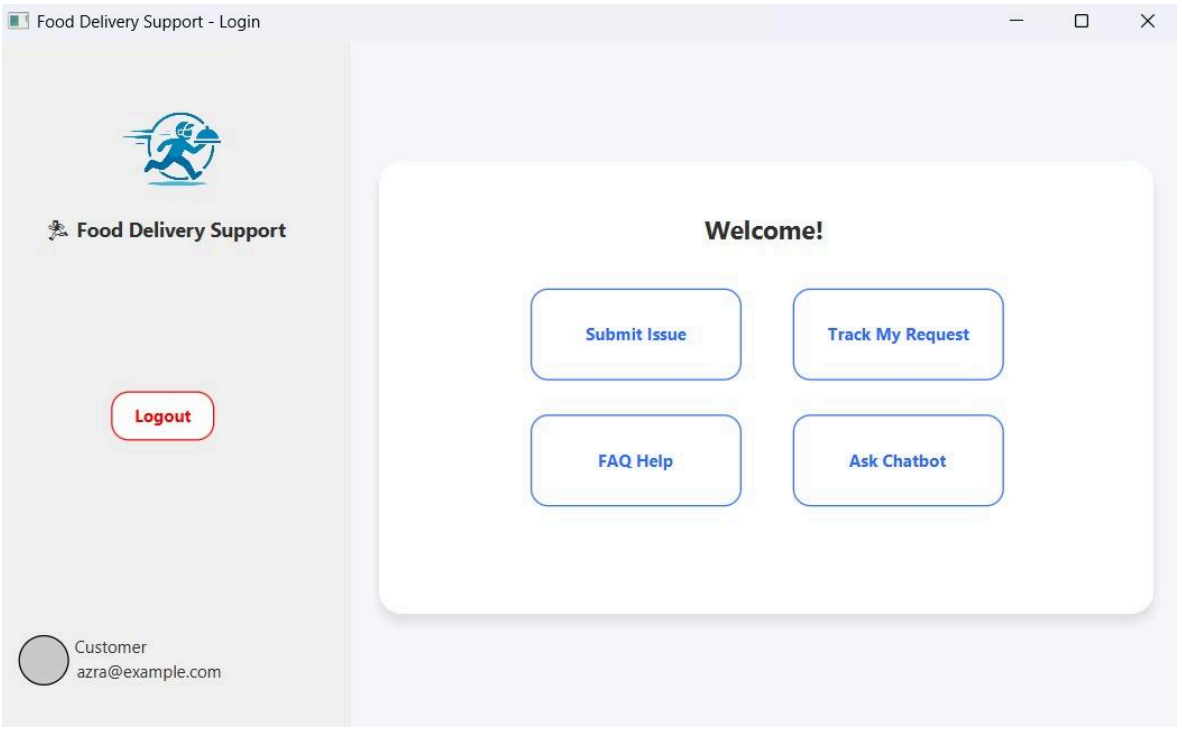


Figure 15: Login with new account

TC05: Submit Issue
Input: Register a new account with valid credentials

- **Category:** Late delivery
- **Order:** #001
- **Description:** my order arrived 2 hours late

Expected Result: Ticket is submitted successfully and Tracked

Actual Result: Ticket created and visible in **"Track My Request"**

Status: pass

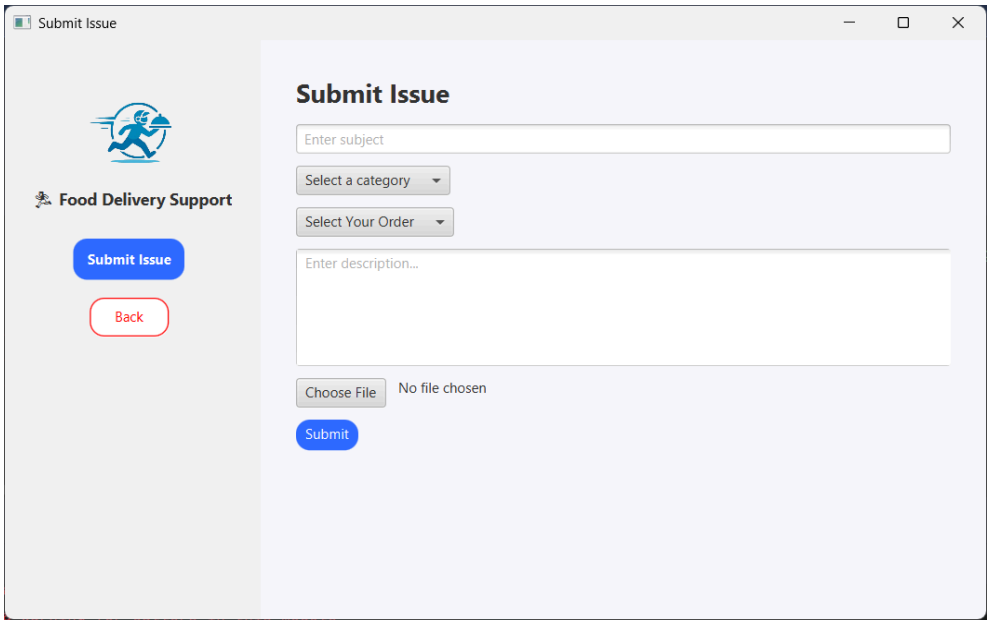


Figure 16: Empty submit issue form

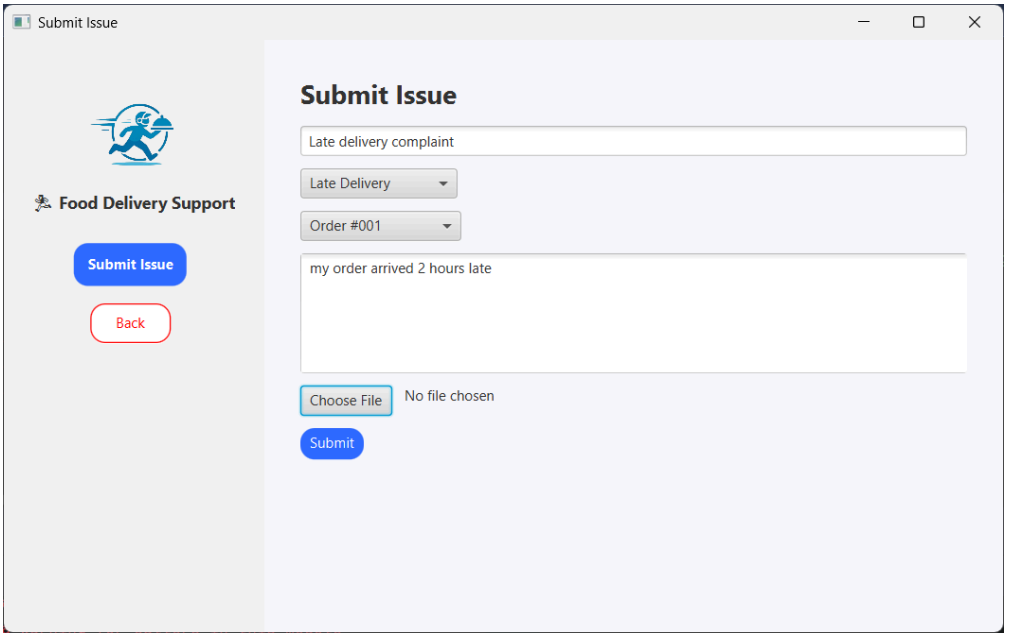


Figure 17: Filled submit issue form

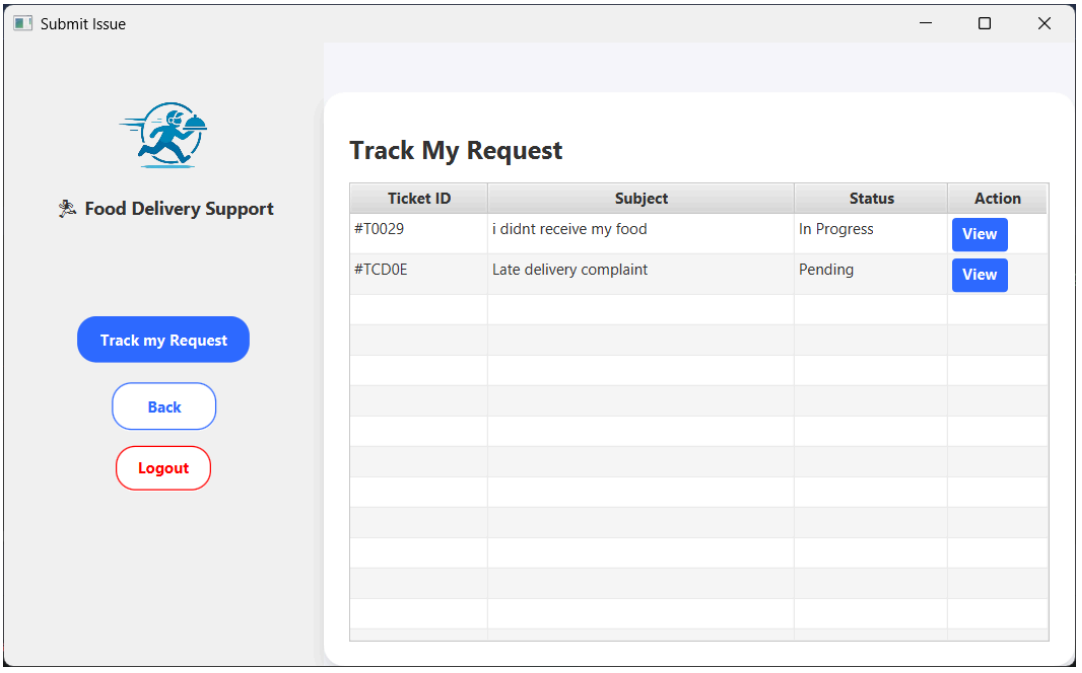


Figure 18: Track My Request

TC07: Chatbot

Input: User opens **Chatbot** and sends different types of messages

Expected Result: Chatbot provides the users with predetermined answers and reacts to the pressing of buttons.

Actual Result: Chatbot worked quickly and every button was responsive as intended.

Status: pass

Observation:

This part presents reports on a few test cases that were performed to confirm the operations of the system. Not every test case has been carried out; however, the selected ones reflect the essential user interaction and significant application features. Test cases such as some border cases (e.g. edge cases or duplicated flows) have not been tested because of a time pressure problem and the scope, but their functioning has either been logically approached or indirectly exercised through similar situations.

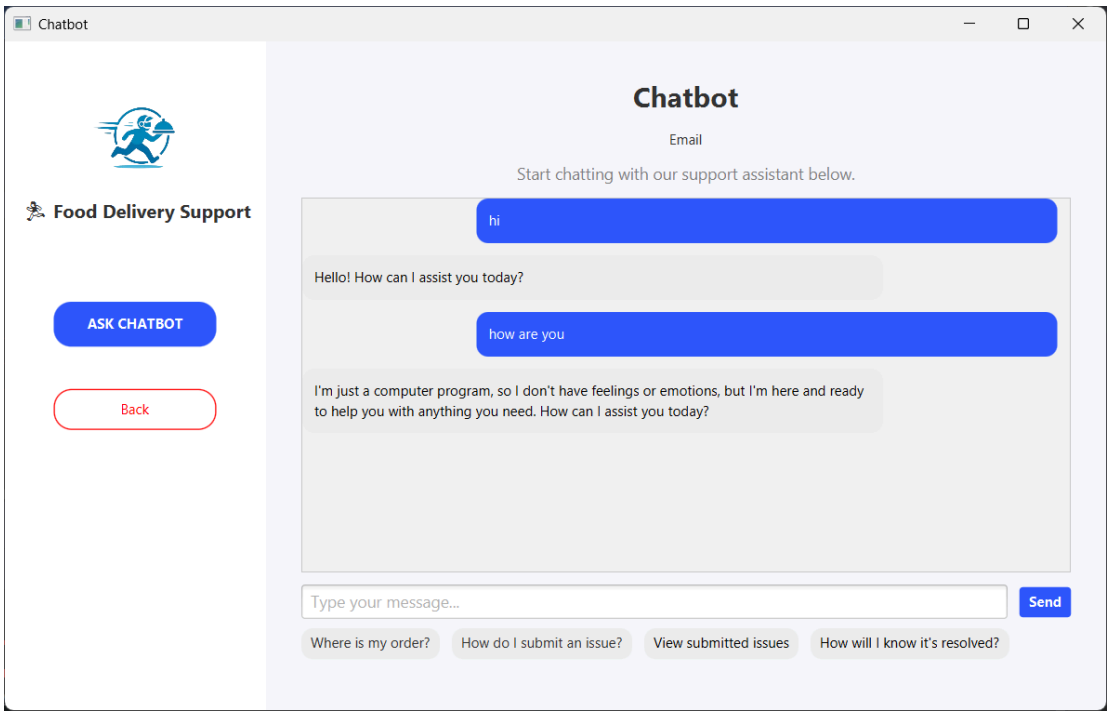


Figure 19: Start chat

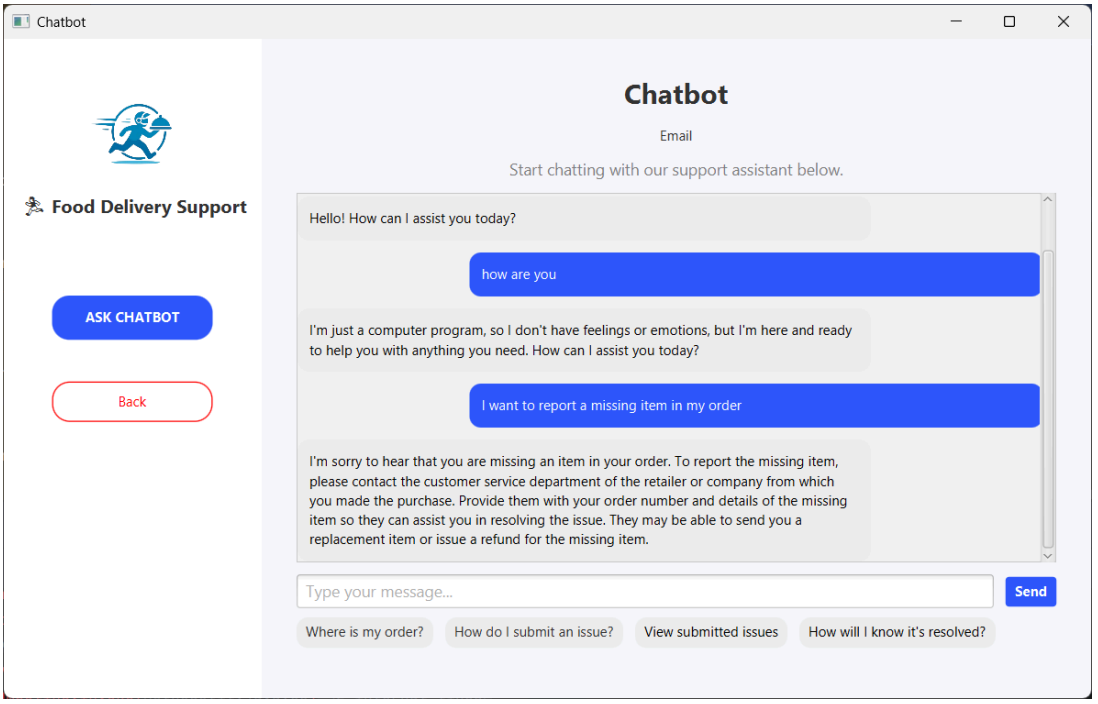


Figure 20: Chatbot automated response

TC09: Admin Login

Input: Register an existing admin account with valid credentials

- **Username:** admin@example.com
- **Password:** adminpass

Expected Result:

- Admin is able to login successfully
- System displays a confirmation message
- Admin is redirected to the Admin Dashboard

Actual Result:

- Message display: Login successful as admin
- Admin was redirected to the Admin Dashboard

Status: pass

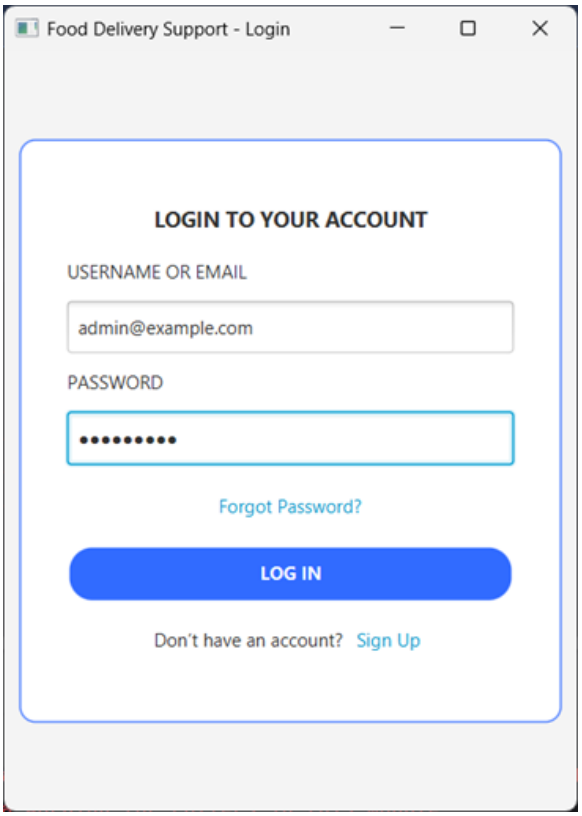


Figure 21: Login as admin

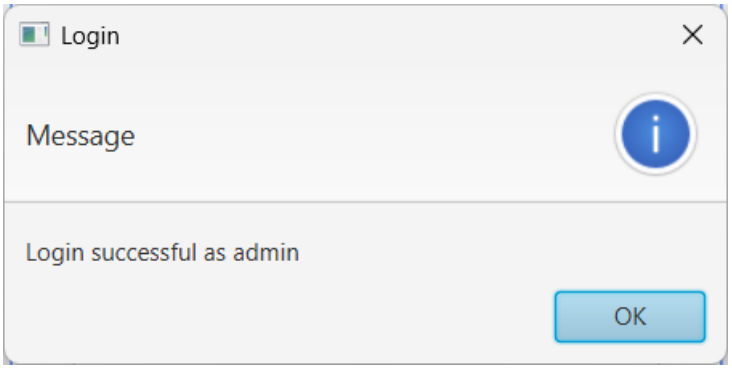


Figure 22: Login success

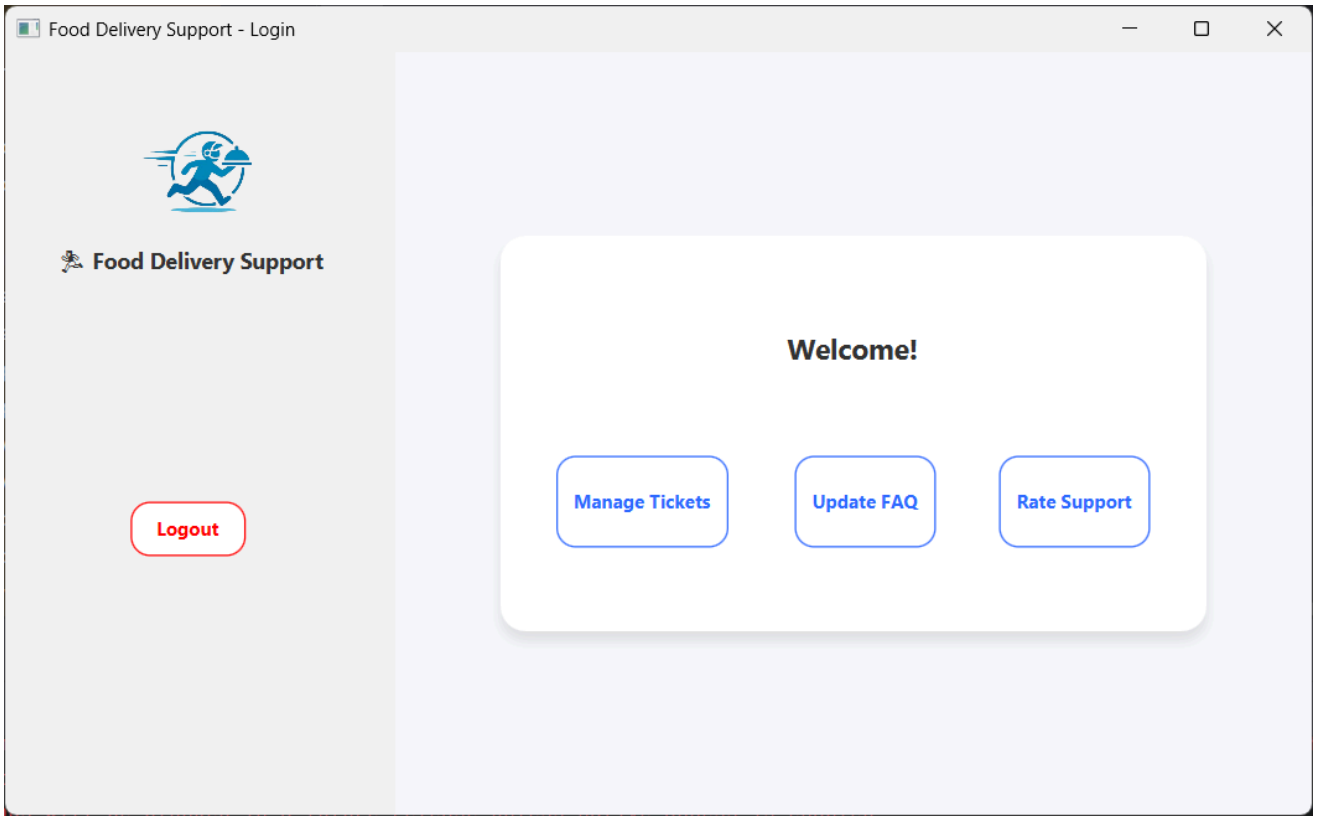


Figure 23: Admin dashboard

TC10: Manage Tickets

Input: Admin logs in and clicks **“Manage Tickets”**

Expected Result:

- Admin can view,update,and respond to support tickets.

Actual Result:

- Admin successful accessed the ticket list, opened ticket #t0029,and respond using auto- reply and status update options.

Status: pass

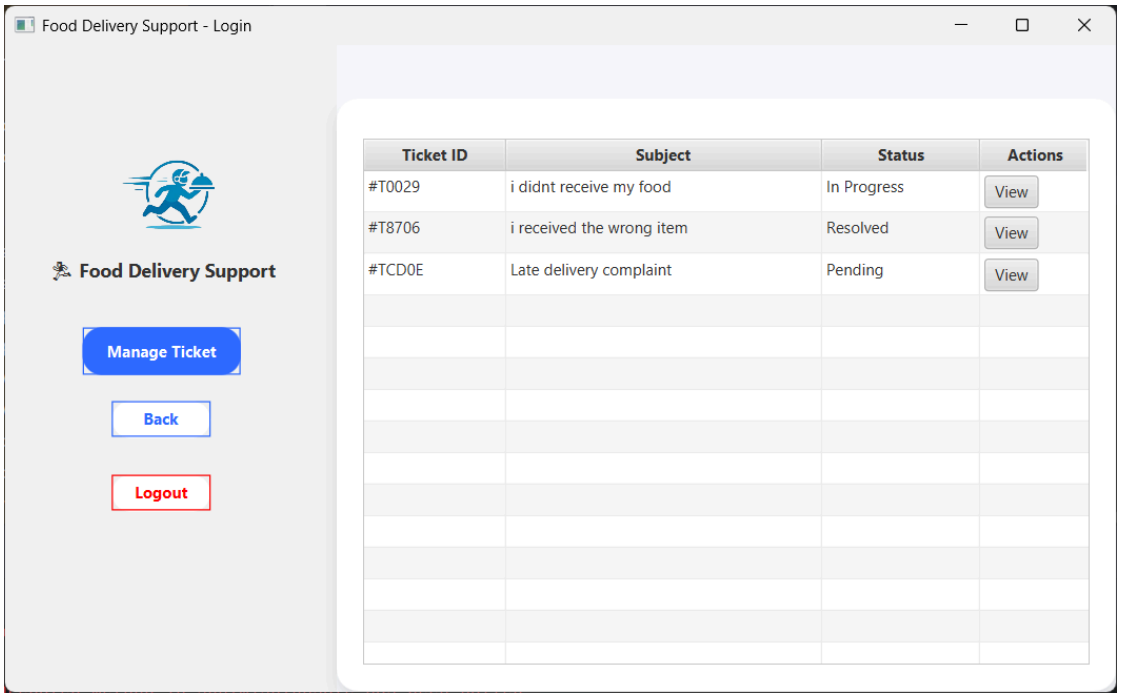


Figure 24: View and respond to ticket

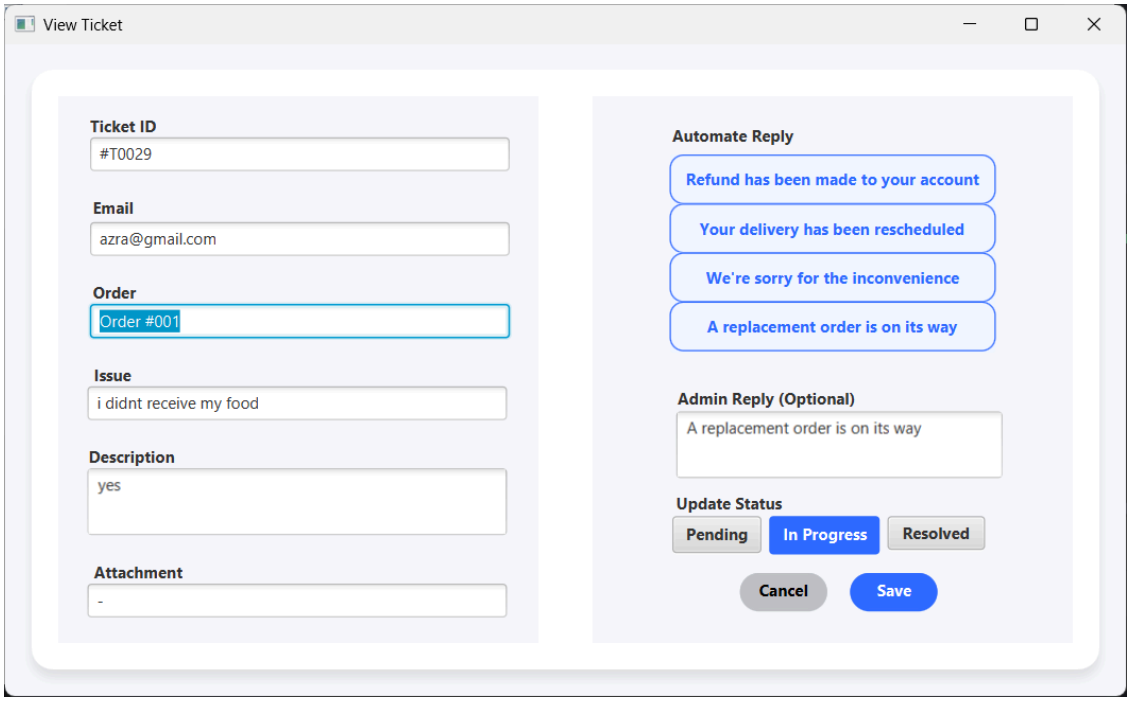


Figure 25: Ticket detail view and admin actions

TC11: Update FAQ

Input: Admin and adds FAQ entries

Expected Result:

- Existing FAQs can be edited.
- New FAQ entries can be added successfully.

Actual Result:

- Admin was able to modify existing questions.
- New entries were added and displayed correctly.

Status: pass

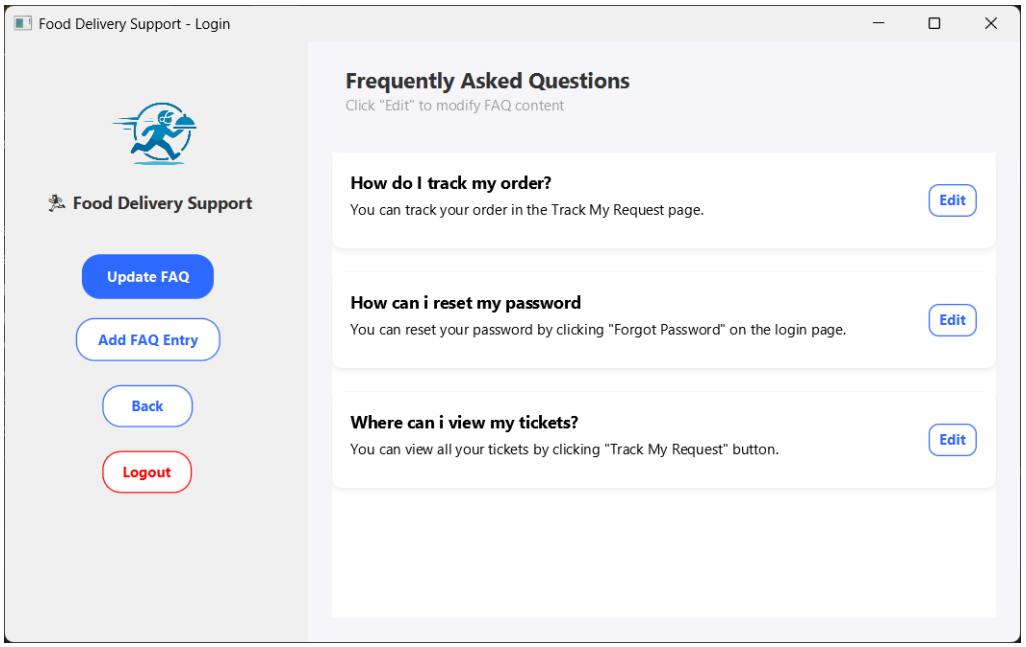


Figure 26: FAQ management

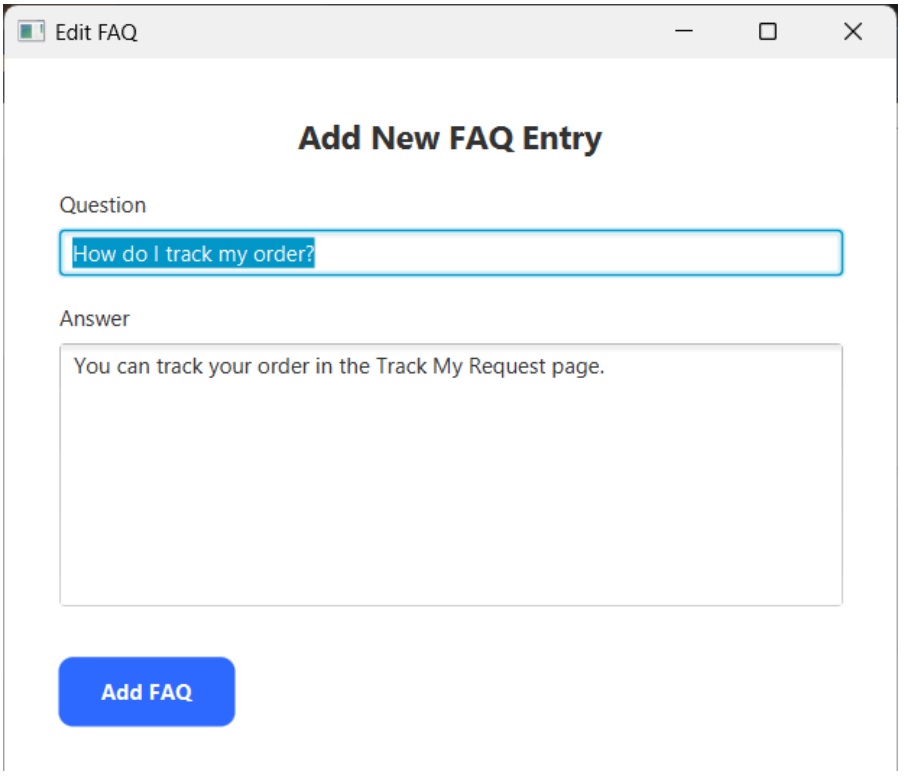


Figure 27: Editing an existing FAQ entry

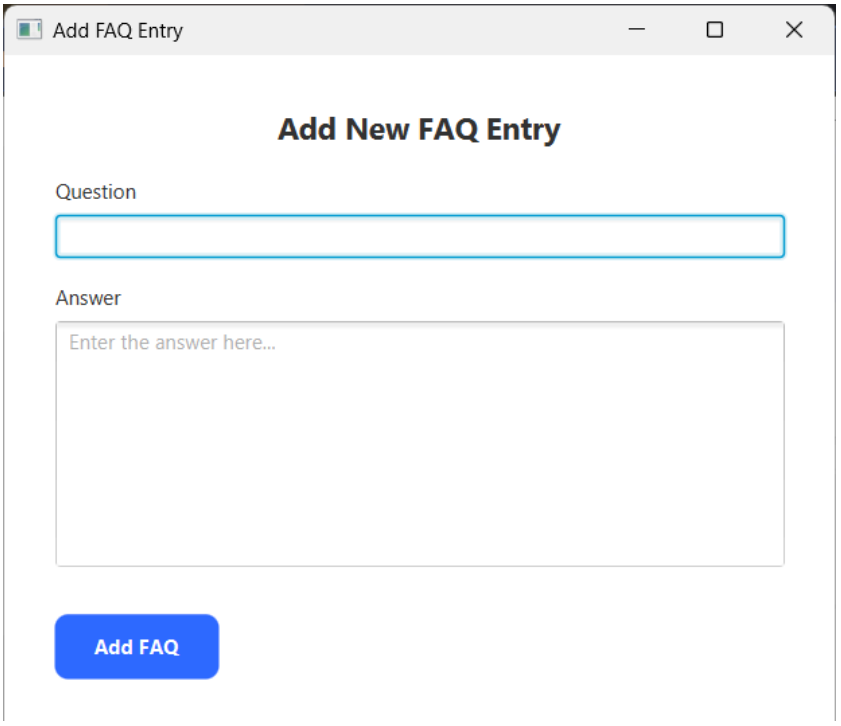


Figure 28: Adding a new FAQ entry

2.3 UI Wireframes Referenced in Test Cases

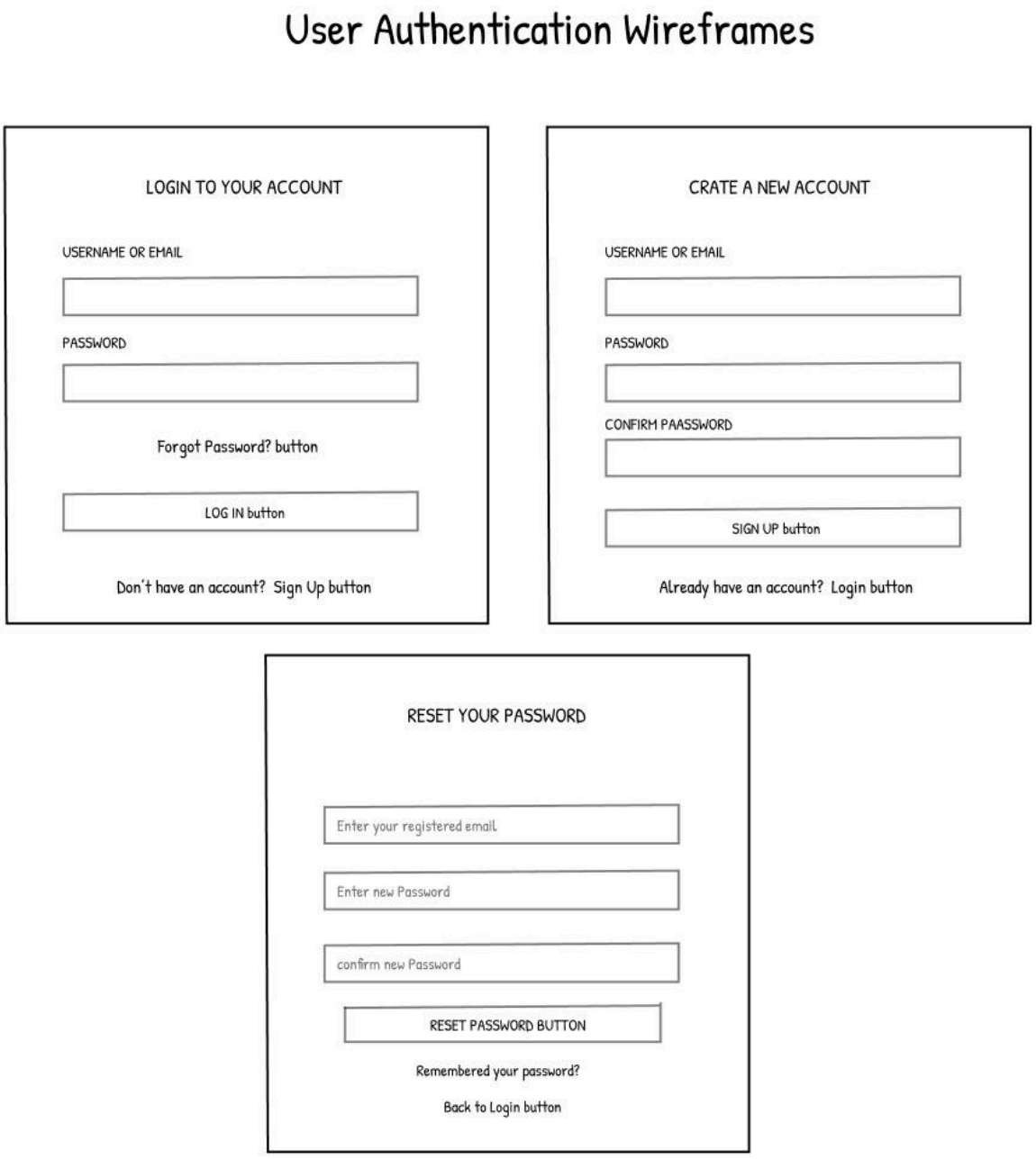


Figure 29: Authentication Wireframes

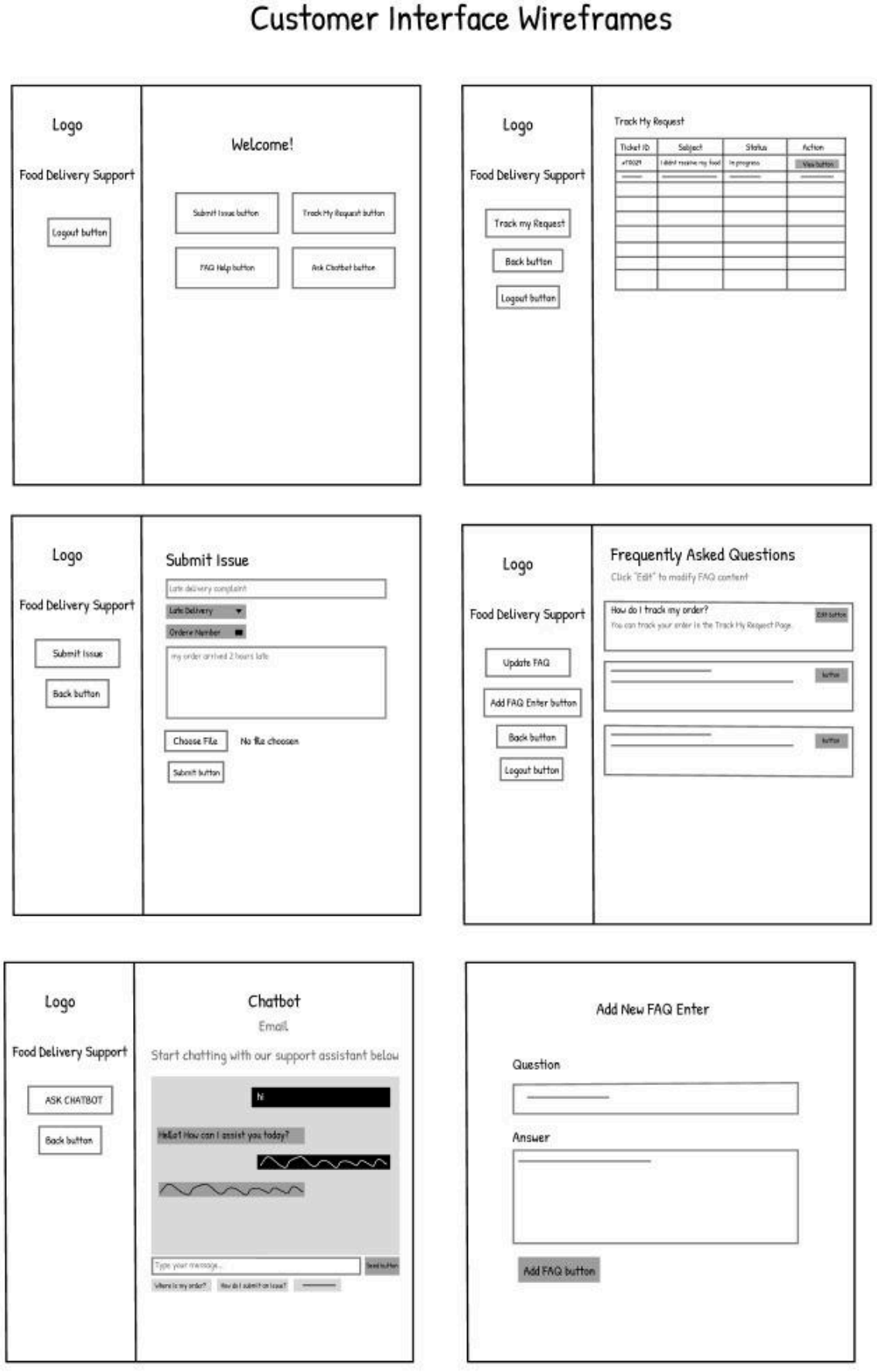


Figure 30: Customer Interface Wireframes

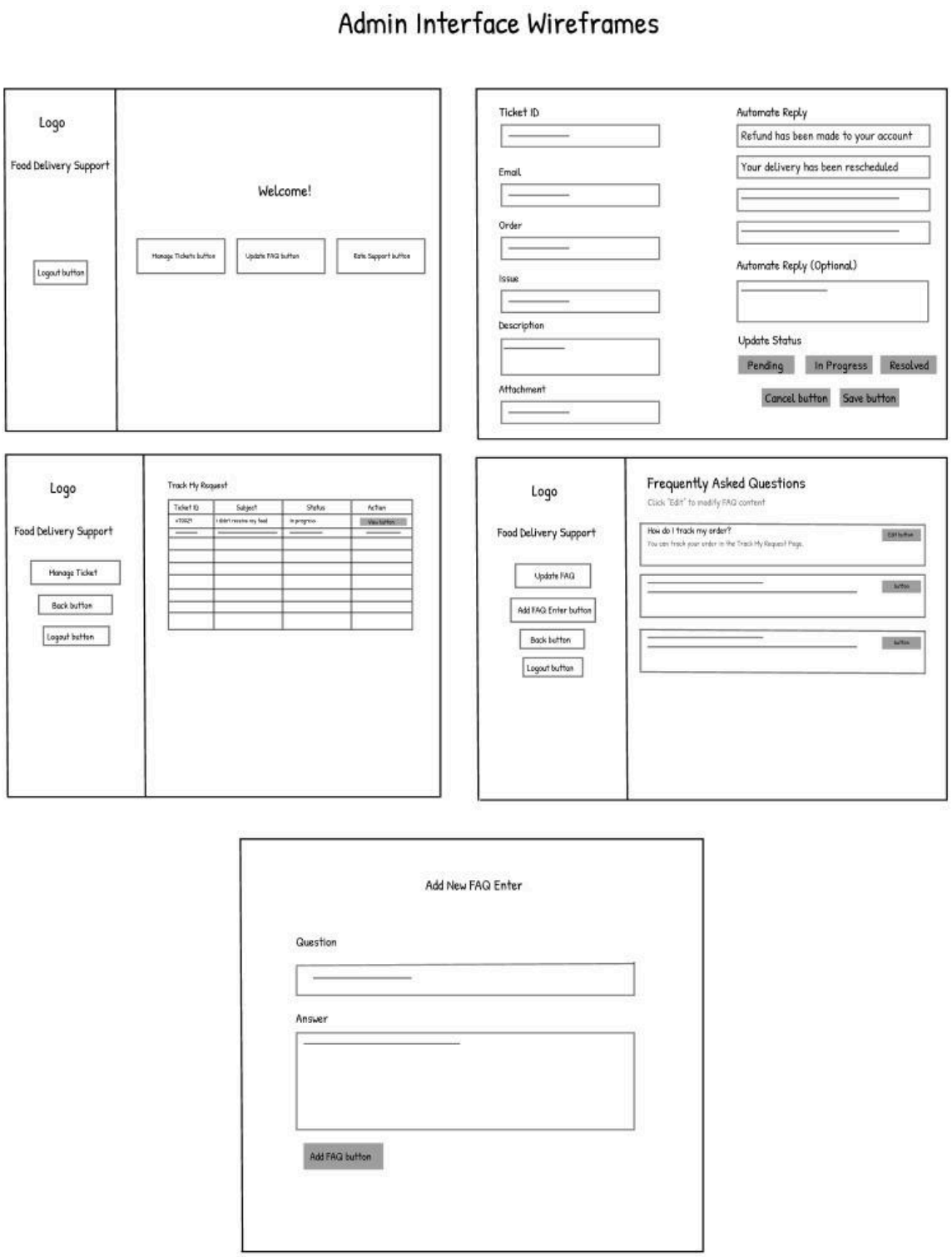


Figure 31: Admin Interface Wireframes

2.4 Wireframe Descriptions

In order to facilitate the testing procedure, briefly, all the interfaces tested will be outlined as given below under two categories: user type and functionality.

A. Authentication Wireframes

1. Login Page

Gives users an option to log in with email and passwords registered. Contain the buttons: forgot password?, log in, a registration button.

2. Register Page

Allows creating a new account by supplying email, password and confirming it. Has a create account button, and a link leading to the log in.

3. Reset Password Page

It enables users to change their password through registered email. These are email, new password, and confirm password. Concludes with a reset link and a link that brings one to the log in position.

B. Customer Interface Wireframes

1. Customer Dashboard

Main navigation screen with buttons that can be used to submit the problem, follow requests, open a chatbot or read FAQ. Logout available.

2. Submit Issue

Gives the customers the ability to write about a problem, make an order, and possibly attach a document. Has a back and submit button.

3. Track My Request

Shows a table of the submitted tickets that have such fields as Ticket ID, Subject, Status, and Action (e.g., View).

4. Frequently Asked questions

Offers an answerable and searchable list of questions and answers. Has the possibility to go back, logout, or add new FAQs.

C. Admin Interface Wireframes

1. Admin Dashboard

Admin login page including access to ticket administration, answer to frequently asked questions, and ticket support responses.

2. Manage Tickets

Displays advanced view of ticket with field as Ticket ID, Email, Order, Issue, Description, Attachment. Helps admins change status and use auto-replies.

3. Track My Request

Manager view of all tickets, just like the customer view except he is the manager.

4. FAQs Management

Enables modifications and maintenance of questions and answers. Buttons to know how to replace or add new FAQs.

5. Add New FAQ Enter

The form to add a new question and answer is simple. Completes with an Add FAQ button.

3.0 User Guide / READMEl

3.1 Introduction

This user guide will assist users to navigate and learn to use the Food Delivery Support System which is made using JavaFX. The aim of the system is to give customers a platform to report their issues with food delivery like delivery delays, missing food items and so on in an easy way. It enables users to make support tickets and contact with administrators and prompt help through intertwined chat robot towards an inquiry and answers area. To administrators, the system also presents features that allow them to view, process and reply to the tickets filled using a specific dashboard format. This guide aims at guiding users through the key features of the system as well as securing an excellent entry into the use of the application.

3.2 User Accounts

NO	Username or Email	Password	Role
1	admin@example.com	adminpass	Administrator
2	azra@gmail.com	azra123	Customer
3	aisyah@gmail.com	aisyah123	Customer

Note:

- These credentials are used for testing purposes during system demonstration and development.
- After using Reset Password, the old password becomes invalid. Only the new password is accepted for login.

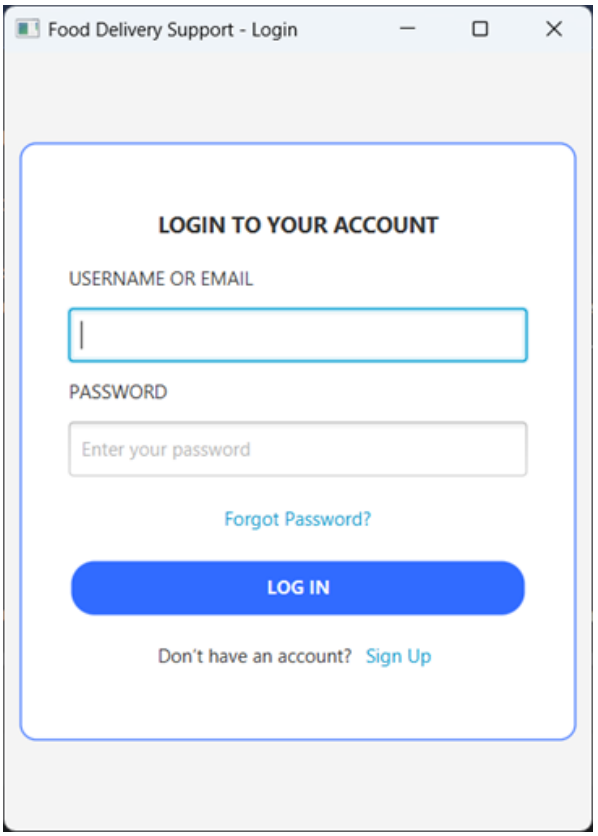
3.3 System UI Walkthrough

3.3.1 Customer Interfaces

3.3.1.1 Login Page

Once the Food Delivery Support System has been launched, this is followed up by the information displaying login information to the users. The screen is conceived in a way that preaches minimalism and simplicity in order to have an easy experience even by returning users and first-time users.

The interface contains a minimum of necessary elements of logging in and shows immediate access to the recovery of the password and registration of the account. It has a straightforward design and a user-friendly structure that makes a user skip to their accounts easily and reliably.

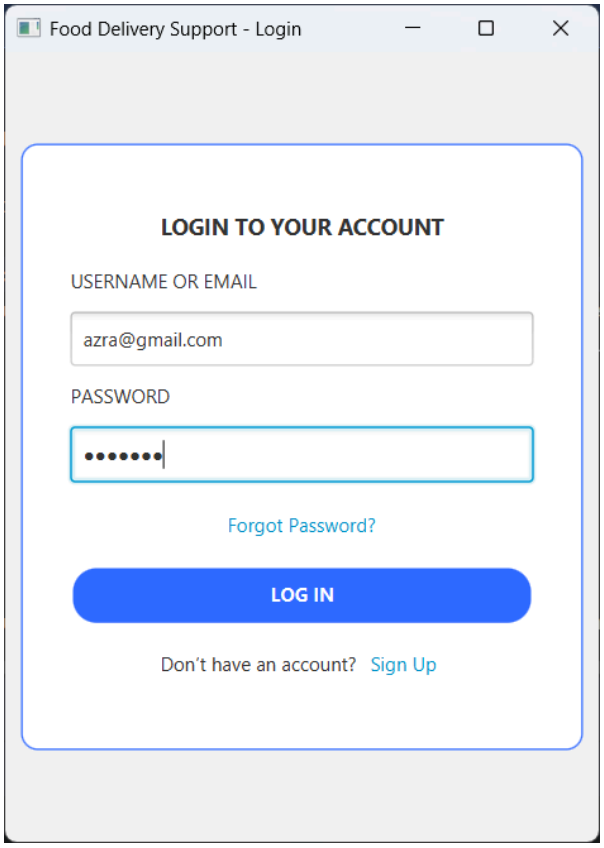


3.3.1.2 Customer Login Validation

This screen shows a normal login attempt with a user providing his/her correct credentials. As depicted by the picture, the customer enters a registered email address (azra@gmail.com) and the respective password. After clicking on the LOGIN button, the computer screens the given information with the stored information in the user file.

When a match is found in the credentials the customer gets access to the customer dashboard. But in case of mismatch, an error message of incorrect login details shall be intimated to the user.

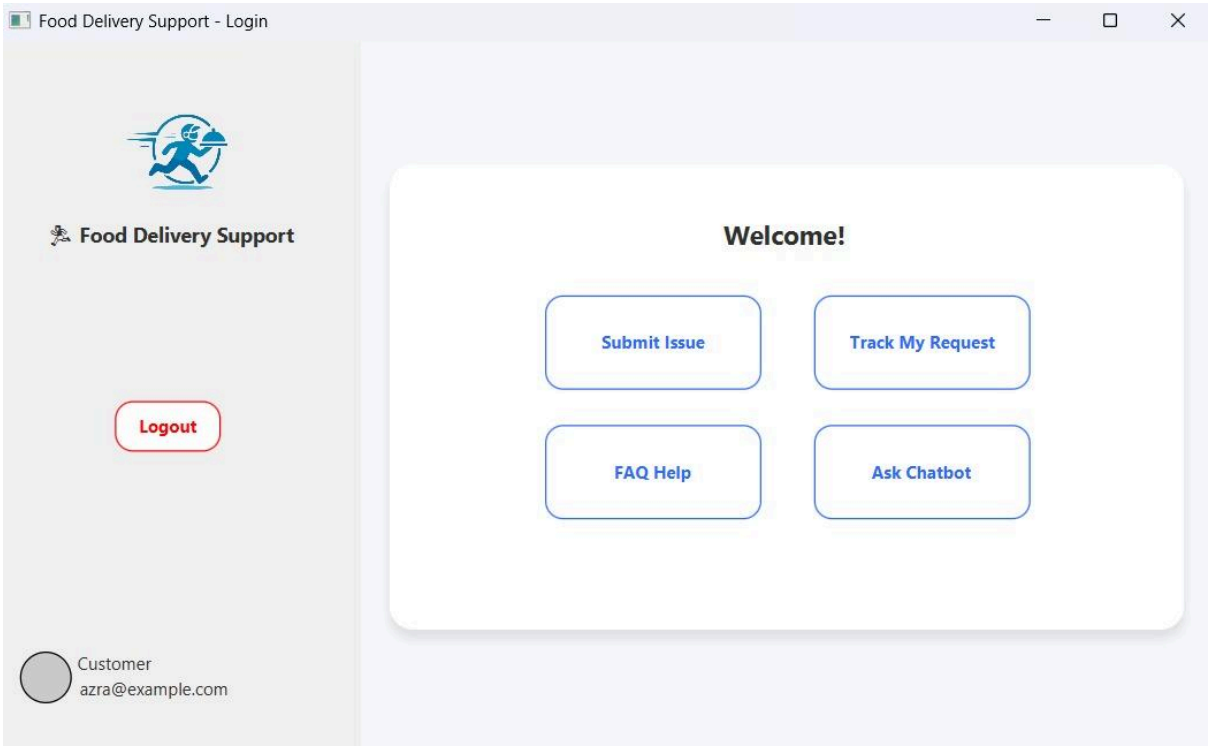
This page makes the process of logging in idyllic by giving instant feedback and secure validation thus, in a case, where a valid user attempts to log in he or she can do so using their valid information.



3.3.1.3 Customer Dashboard

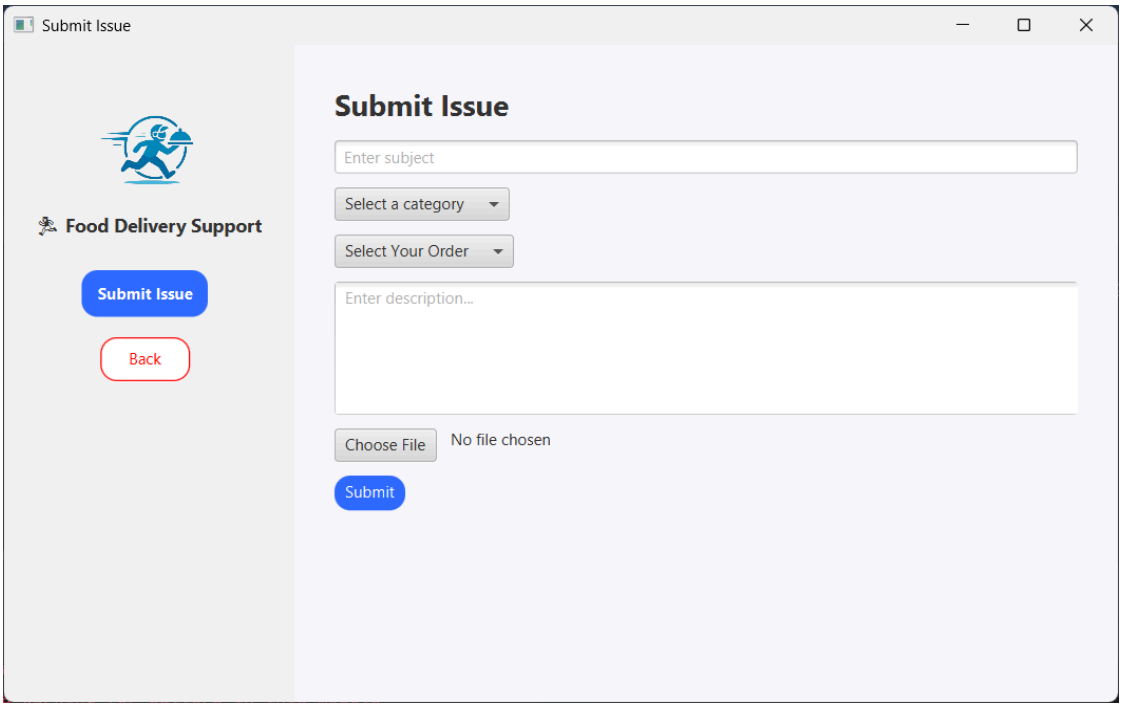
After the user logs into the application, he/she is taken to the Customer dashboard. It also displays a greeting message and provides access to the most significant functions of the system that involves a report of the issue, request tracking, help through FAQ, and interaction with the support chatbot in a conversation. And, of course, there is an opportunity to log out which also takes little time to access in the sidebar.

The design is done in a manner that the structure easily walks in and is largely comprehensible and the support elements are client accessible.



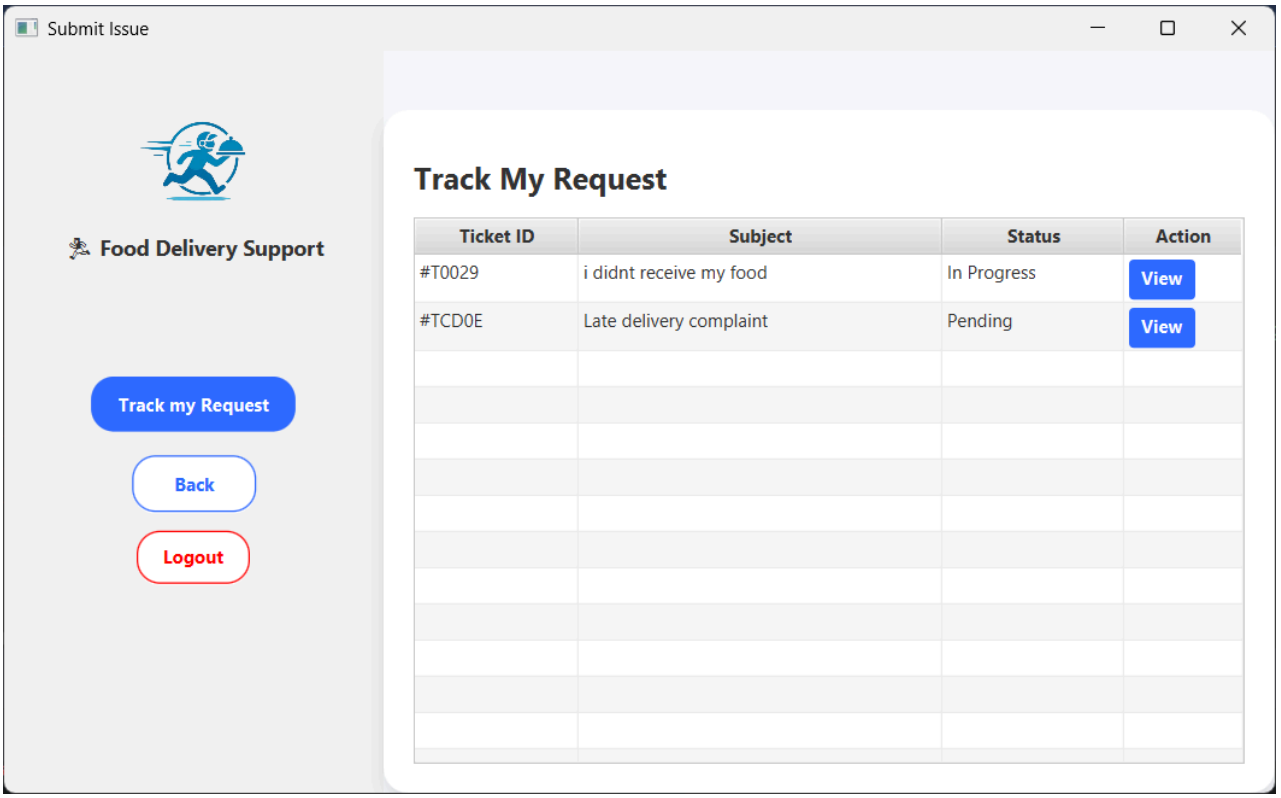
3.3.1.4 Submit Issue

Once the user lands on the Submit Issue page, they can report issues regarding their orders of food in an easy and systematic manner. It will allow them to type in a subject title, select an issue category in a dropdown list, and pick the particular order in it. The problem can be described within a description box, and a user also has an opportunity to add a file as additional evidence. After all the fields are entered, a user is allowed to press the Submit button. On this page, problems are well elaborated, but also supported, which allows the support team to solve them more quickly.



3.3.1.5 Track My Request

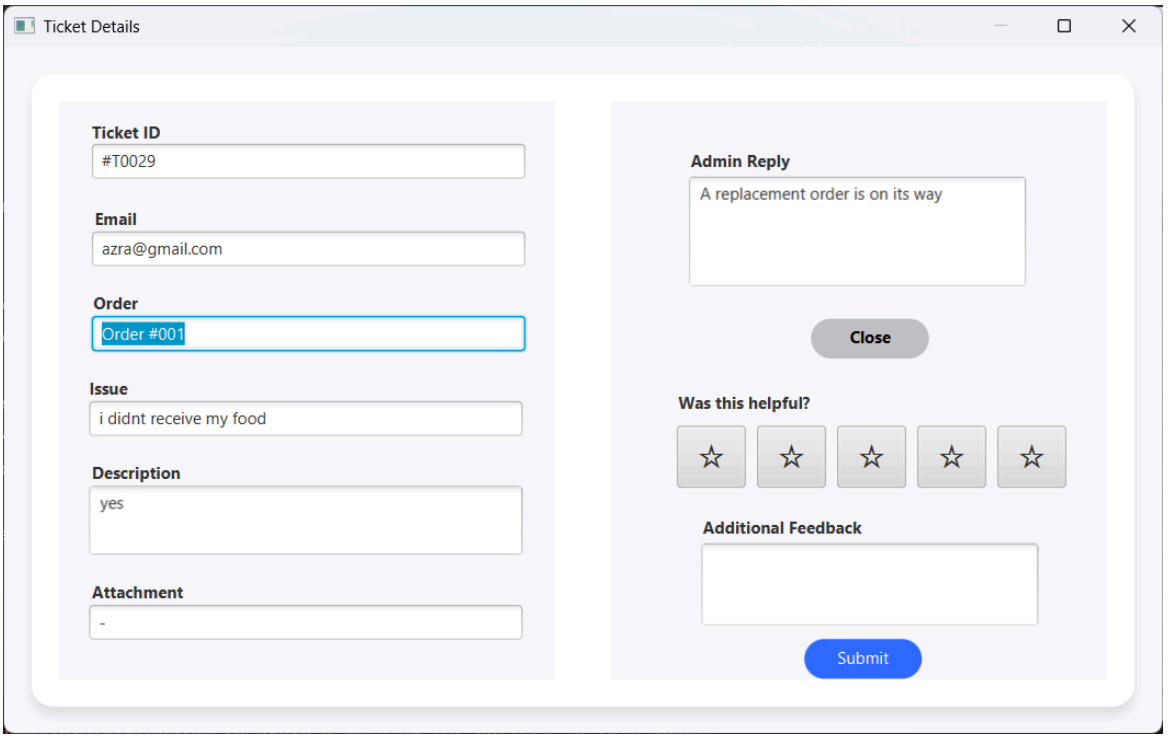
Customers can also check their status of a request using this page after making an issue. Every ticket is displayed with the id, topic and the current status. By clicking the button "View" the user can look at the most recent updates or comments of the support team. This webpage will make the users aware of the problem and make the users aware that their problems are being actively dealt with.



3.3.1.6 Ticket Details

All the data of a given support ticket are presented in this page along with its content and the response given by the admin. Users will have a chance to see the information about their request they made and assess any reply they get to their request as well.

There is also an interface where user can rate the quality of support distribution system by using stars and give further feedback in text box. When this is done, the user is able to present the feedback on the page. The screen promotes the exchange of information, and the service staff can base their improved service on the opinions of the users.

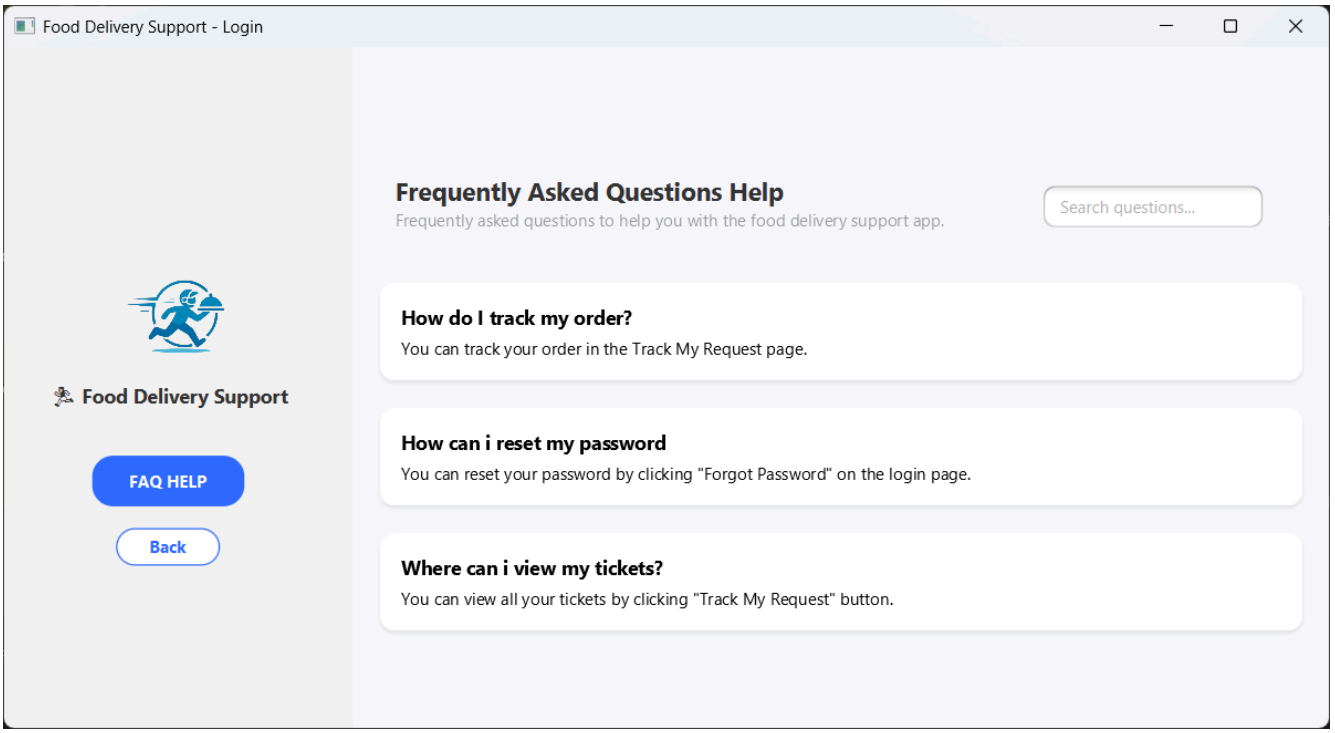


3.3.1.7 FAQ Help

The users can get the help of quickly browsing through the frequently asked questions about the food delivery support system on this screen. All the FAQs have a clean format which resembles a card having a bold statement of the question and a short, useful response.

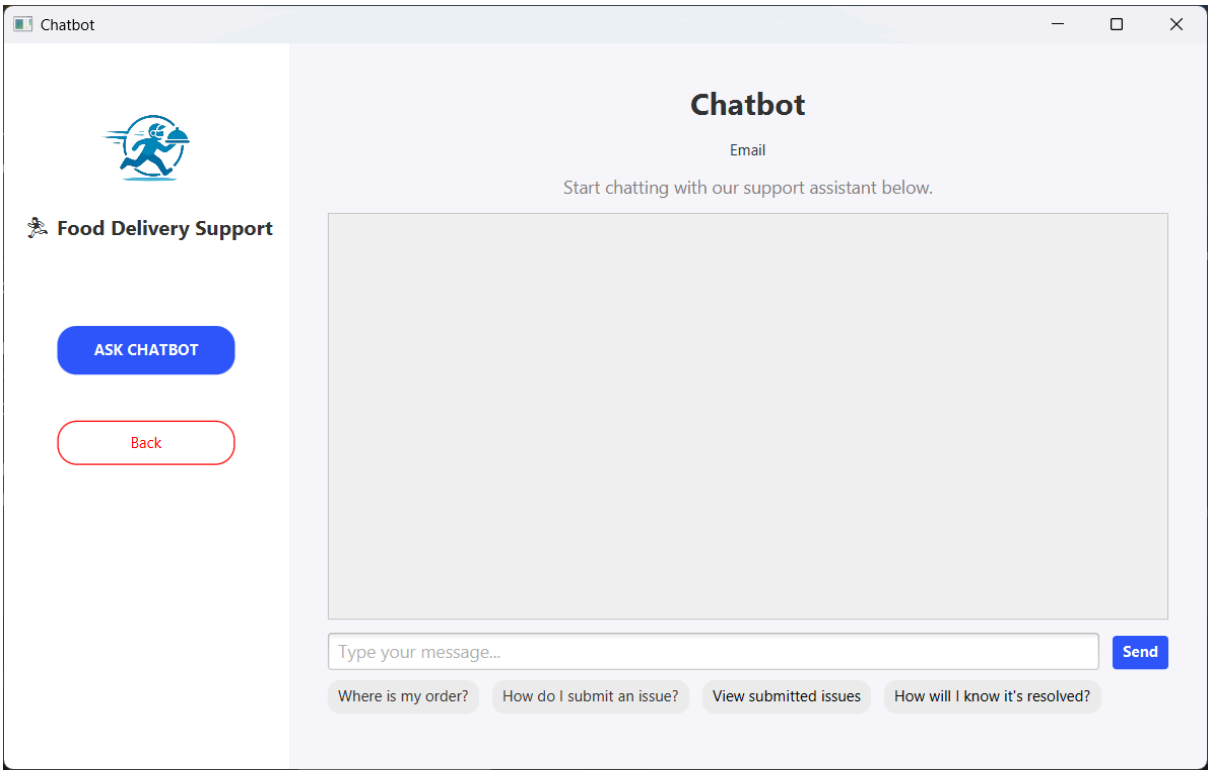
The idea is to cut the lines on support by enabling the user to resolve typical issues independently: e.g. underline the order status, reset a password, look up past tickets. There is also the option of searching by typing in a keyword and a search bar which helps to go through the results easily and a sidebar to move back to the main dashboard.

This design enables the user to solve their problems without the assistance of human agents hence gaining quick access to support.



3.3.1.8 Chatbot

Through this screen, the users can use the automated chatbot assistant to get instant answers to their concerns about their food deliveries. The chatbot will have a text box on the interface where the customers are allowed to type their questions and click on the Send button to access assistance, as demonstrated in the interface. There are also pre-designed short-cut buttons on the bottom including, Where is my order?, How to raise a problem, etc. These possibilities allow providing the users with assistance within a shorter period of time without required typing. This option makes the process of support more efficient and does not require the direct intervention of staff. The sidebar has a Back button that enables the user to come back to the dashboard with ease.

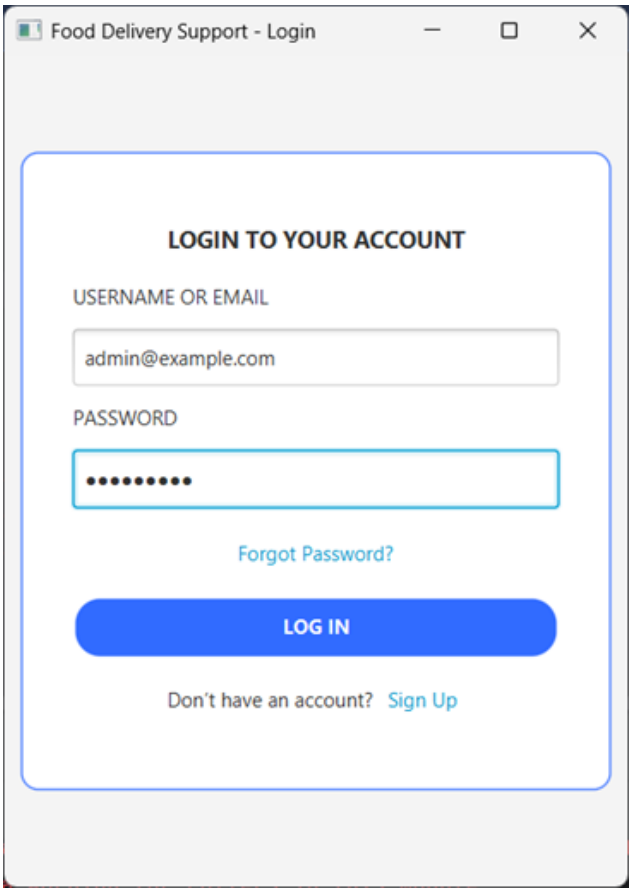


3.3.2 Admin Interfaces

3.3.2.1 Admin Login

This is a screen displaying log in to administrators. Similar to the customer log in page, the administrator also enters his or her log in details, which in this case is the email address of our administrator which is the address, admin@example.com and the password which is adminpass.
After clicking upon the login button, the system confirms the details and gives access to the Admin Dashboard. This will enable the administrator to address support request, track user activities and react to them.

Its structure and design are the same as of a customer login screen, and the set of features and access vary depending on a logged-in user role. The view of the administrator functions is only available upon logging in by an administrator.



3.3.2.2 Admin Dashboard

This is the default control center of the administrator upon successful login. It has three main

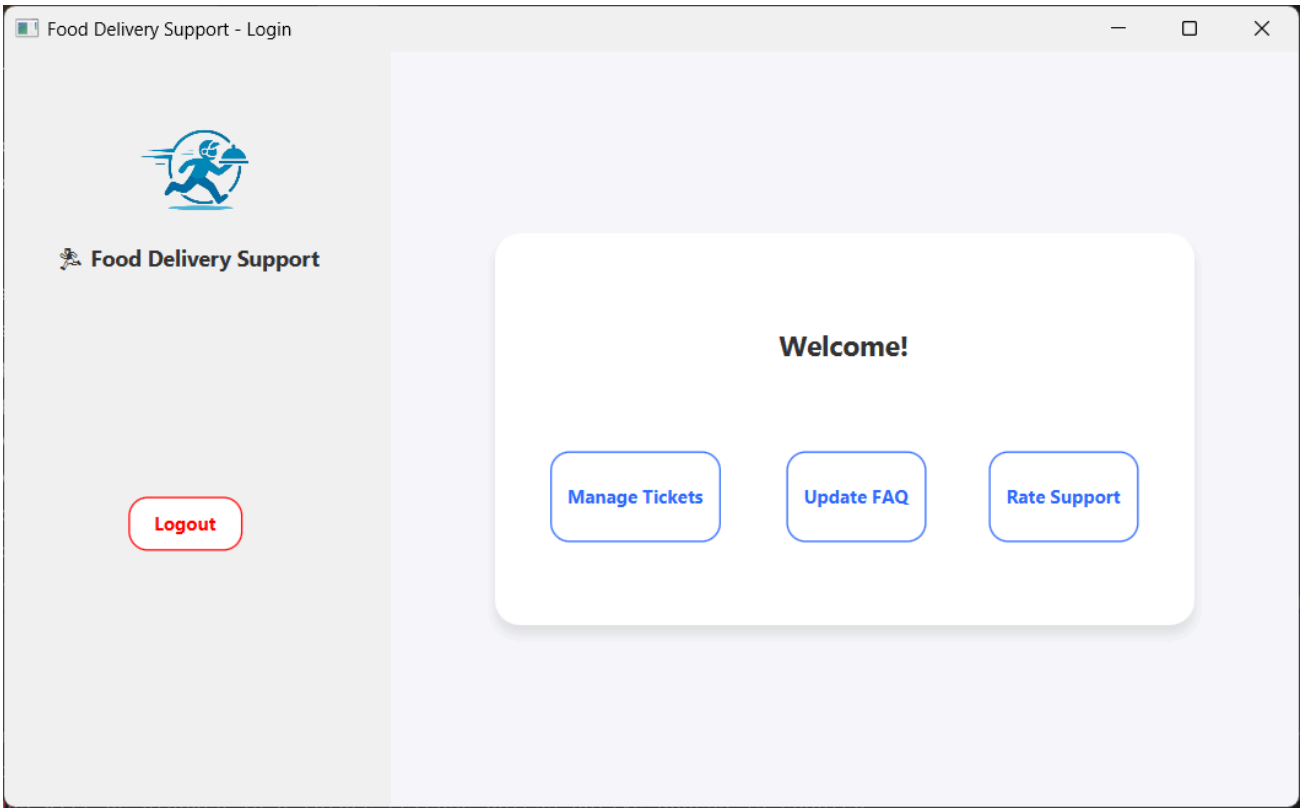
buttons of action:

Manage tickets: Will open the interface in order to examine the tickets submitted by the customers and respond to them.

Update FAQ: Allows the ability to update or create frequently asked questions to better support the customers.

Rate Support: This is a feature that enables one to see the user rating and feedback to understand the support experience.

Signing out is also secure as there is the Logout button on the sidebar. The scheme of the layout is aimed at the rapid response to essential administrative operations.

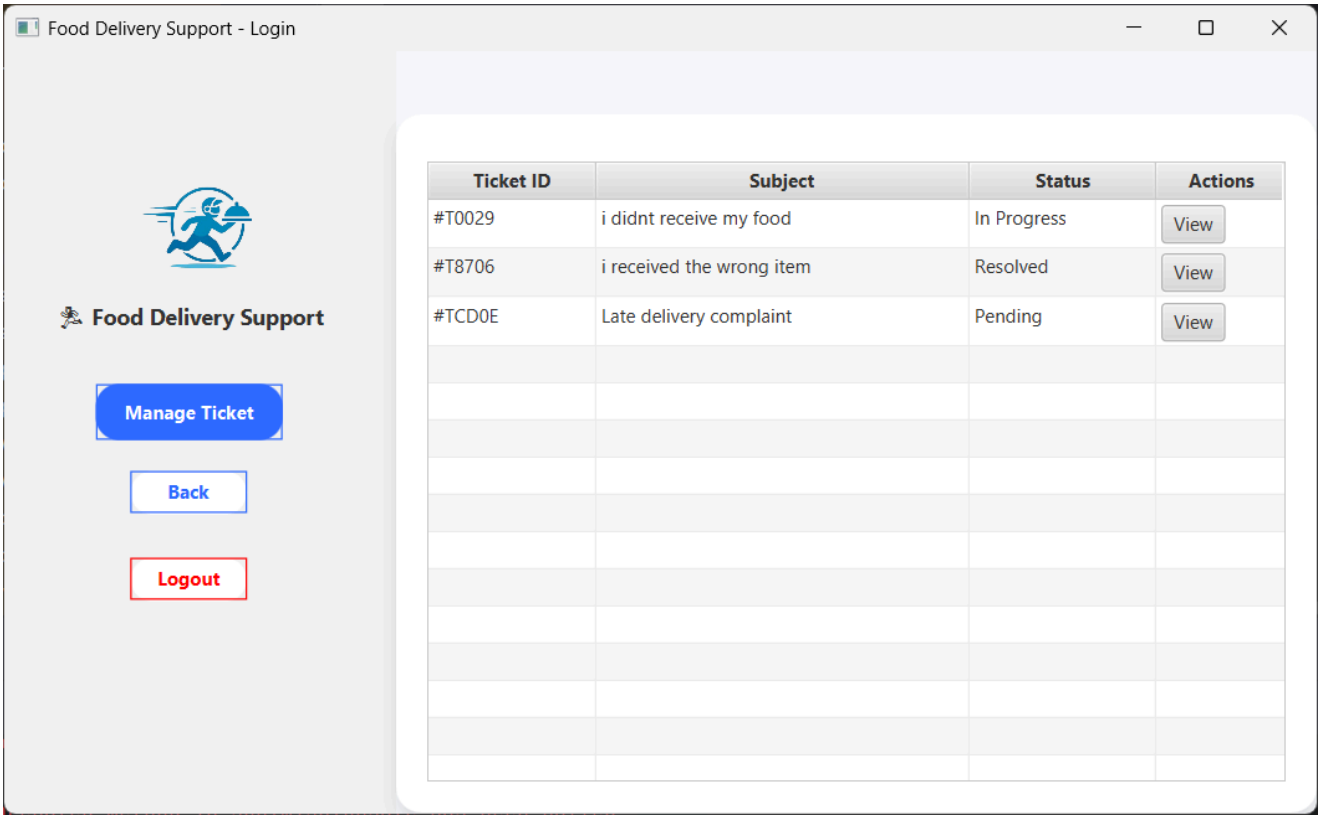


3.3.2.3 Manage Tickets

On clicking Manage Ticket on the admin dashboard, this interface appears showing a table with all the support tickets that have been raised.

The table contains the most necessary information, i.e. Ticket ID, Subject, Current Status, and a View button to review each ticket separately and reply to it as well.

This page provides the admin with the ability to monitor and manage all user complaints according to their current status, and thus the complaints are handled timely and in an organized manner.



3.3.2.4 View Ticket

This interface enables the admin to manage a specific complaint, which he gets after selecting the complaint using the list of tickets. The left panel indicates the submitted information of the ticket such as ID, email, order, issue, and description. The right panel contains lightning action tools of the admin:

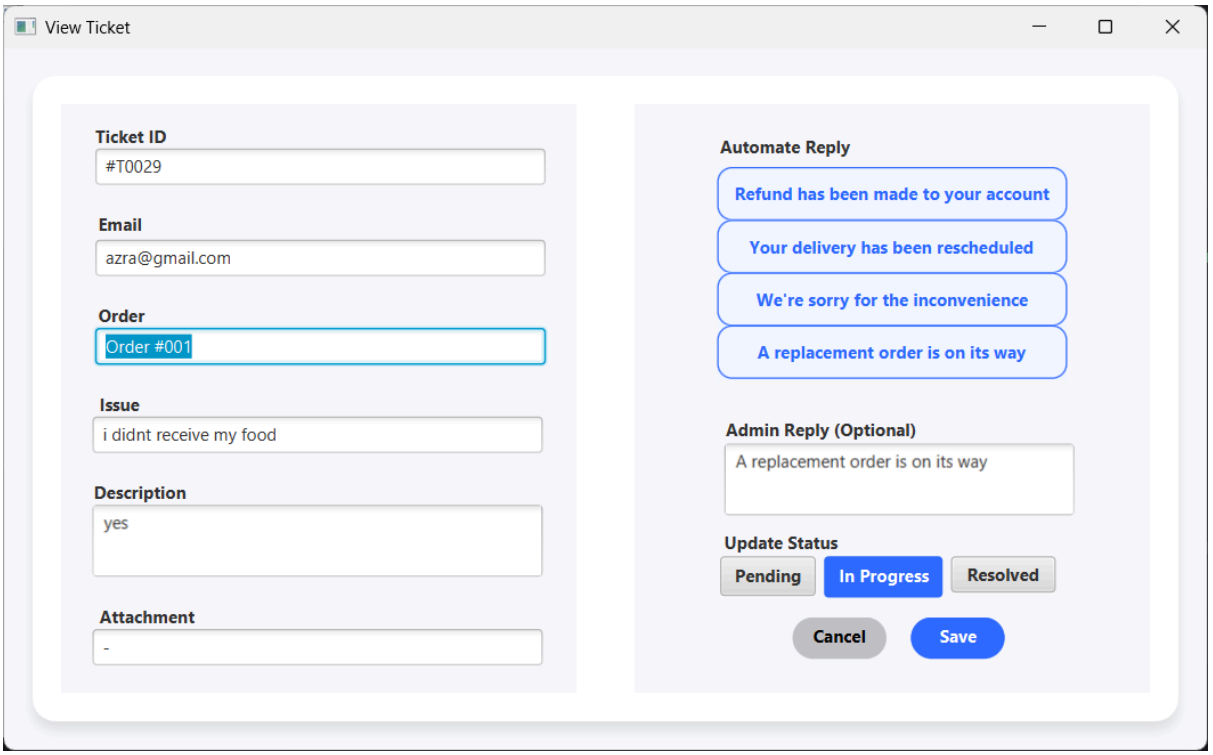
Automate Reply: set messages on frequently asked problems.

Admin Reply: Textarea for a personal message.

Status: Buttons to be able to define the current state (Pending, In Progress, Resolved).

Save / Cancel: Save / Drop changes.

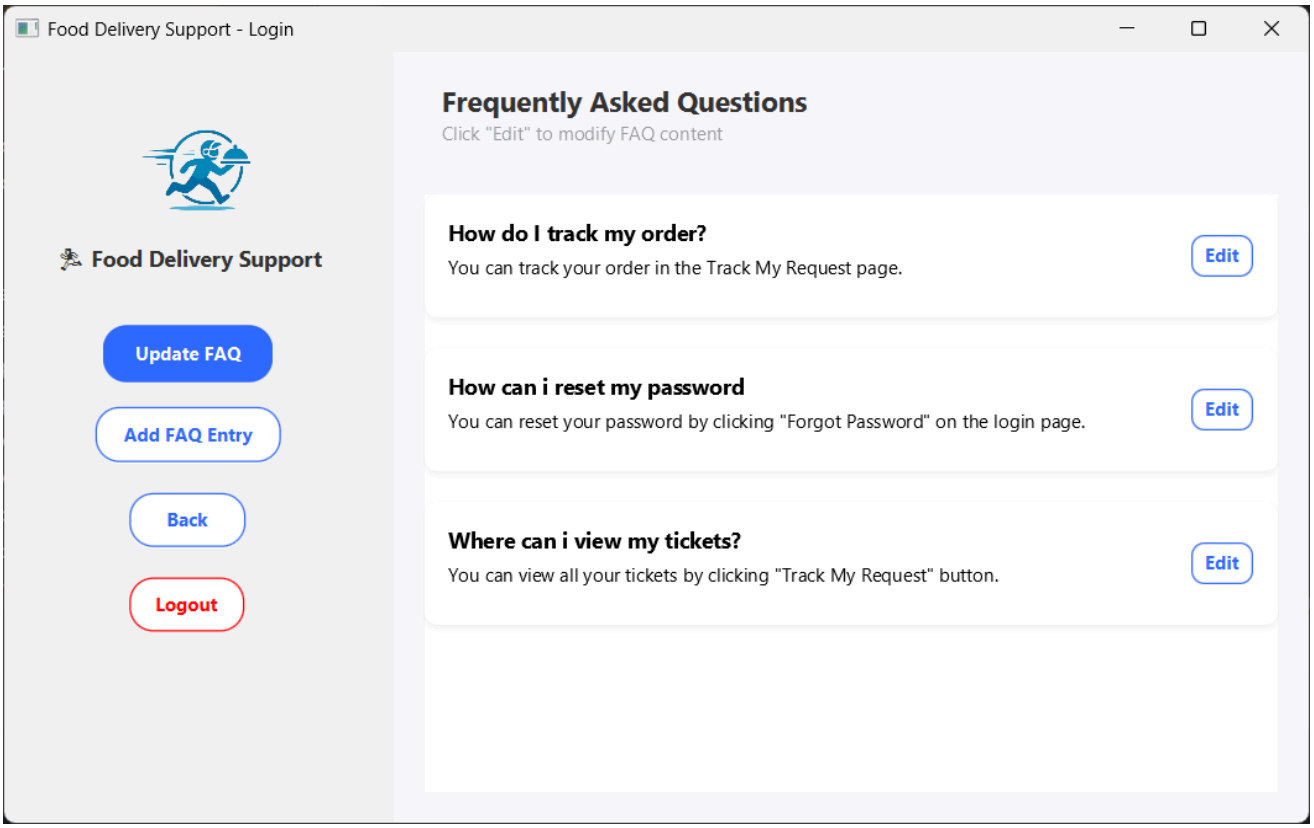
The layout helps in quick resolution of issues and consistency of response.



3.3.2.5 FAQ Update

Through this interface, the admin will be in a position to handle the FAQs of the system. These questions are presented in a form of a card on each question with the question title and its answer. We can easily edit any entry with the given button "Edit" that exists on each card.

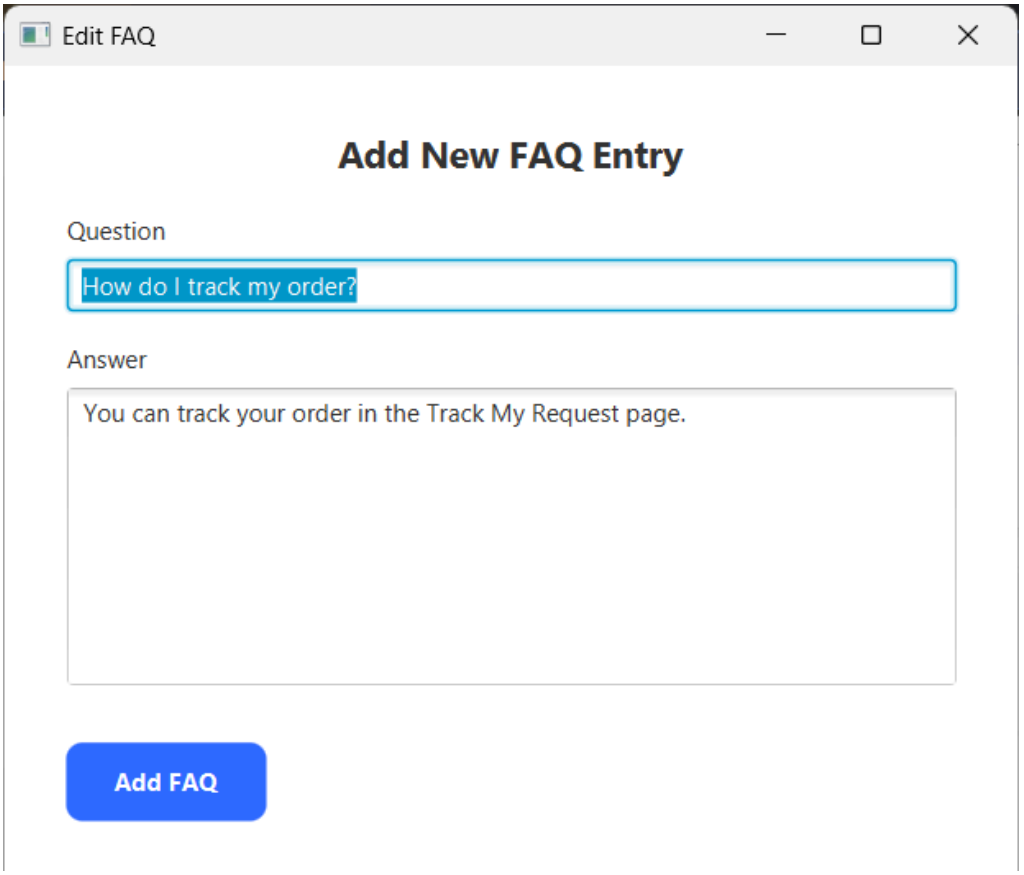
Moreover, any new questions can be inserted using the given form by marking the option of adding the FAQ entry. The structure would allow the admin to make the constant adjustments and edit the content of the FAQs to suit user demands and frequent requests, retaining simplicity and usefulness throughout the support system.



3.3.2.6 Add New FAQ

This screen allows the admin to come up with a new addition on the frequently asked questions. The form has two fields where one may enter the question and to the other to give the answer. When both are done the option to add the entry under the FAQ can be pressed after clicking the button, which reads as Add FAQ.

This is because such a function will enable the content in the support to be dynamically extended and this will enable the system to be responsive to the needs of the user and therefore recurrent questions will be addressed beforehand.



4.0 Conclusion

The completion of this phase produces a successful implementation of the food delivery customer support system, where designs are transformed into a fully functional application built using Javafx and Scene Builder. Core software engineering principles, such as Object-Oriented-Programming , modular design and user centric GUI development has been applied consistently throughout the process. These practices result in a responsive,and maintainable support system specifically in the food industry.

On the administration side, it offers ticket management, FAQ updates, and feedback checks as well through an intuitive neat interface. The responsiveness and usability have been considered, to guarantee an easy experience of the application to the customers and administrator, during the process of navigation and interaction with the system. This feasibility gives an idea of how Java-based desktop application could work in practice and be successfully implemented to address real-life support issues in the food delivery business, which in turn preconditions their future improvements and scalability.

This user guide has taken the customers and the administrators through the critical elements of the Food Delivery Support System. Whether it is reporting the problem and following the tickets or customer support and FAQ updates, the system was usable and efficient. Using this guide, users are expected to have confidence in their ability to navigate through the application, make requests and seek the necessary assistance. Adding these functions, such as chatbot, feedback ranking, and an administrative tool, will also benefit the complete experience and result in the improved delivery of services.

The inclusion of a comprehensive user guide and extensive test coverage further demonstrates the project’s commitment to reliability and maintainability. Through test cases, it helped validate the critical components of the system under various conditions in order to ensure that the application functions correctly and consistently. While the user guide helps the end-user to effectively navigate the application.

In conclusion, the food delivery customer support system represents a practical application based on Javafx technologies in addressing real-world customer service needs. It provides a strong foundation for further enhancement to be made to improve the system. Future iterations could apply cloud-based databases, mobile compatibility, AI-driven support, and multilingual support to enhance its capability.

5.0 Rubrics

Marks	100% of allocated marks	75% of allocated marks	50% of allocated marks	25% allocated marks	0% allocated marks
Criteria	• Complete understanding of the problem • A plan that could lead to a correct solution with no algorithmic errors. • Correct solution	• Misinterprets a minor part of the problem. • Substantially correct solution with minor omission or procedural error	• Misinterprets a major part of the problem. • Partially correct solution but with a major fault • Computational error, partial solution for the problem.	• Completely misinterprets the problem. • Substantially in-appropriate solution	• No attempt • No answer or wrong answer based upon an in-appropriate solution
10	**10**	**7.50**	**5.00**	**2.50**	**0**

6.0 Appendix

Appendix A: Wireframe

https://www.figma.com/design/ZtjTrXpbtz4WBgHZSnIpaZ/Untitled?node-id=0-1&t=6GFpLMKObLpmOXCB-1